

12

EUROPEAN PATENT APPLICATION

21 Application number: **88112881.3**

51 Int. Cl.4: **G06F 15/42**

22 Date of filing: **08.08.88**

30 Priority: **13.08.87 US 85511**

43 Date of publication of application:
22.02.89 Bulletin 89/08

84 Designated Contracting States:
CH DE FR GB LI

71 Applicant: **SYNTHES AG**
Grabenstrasse 15
CH-7002 Chur(CH)

72 Inventor: **Dormond, Kenneth**
30 Deer Run Lane
Malvern, Pennsylvania 19355(US)
Inventor: **Friedman, Robert J.**
1100 West Chester Pike, Apt. D 8
Paoli, Pennsylvania 19382(US)

74 Representative: **Lusuardi, Werther G.**
Dr. Lusuardi AG Stockerstrasse 8
CH-8002 Zürich(CH)

54 **Expert system.**

57 An expert system which provides one or more suggested treatments for a patient with physical trauma is disclosed. The system includes a computing device having a memory, a plurality of data bases in the memory, an application program and an inference engine program. The data bases include graphical illustrations of different types of physical trauma, and a knowledge base which contains treatment information. The application program is executed in the computing device and interactively displays a series of screens including at least some of the graphical illustrations, to elicit responses from the user concerning the specific type of physical trauma and specific characteristics of the patient. The inference engine program, which is also executed in the computing device, uses the knowledge base and information related to the responses elicited from the user, for selecting one or more suggested treatments. The application program presents the suggested treatments to the user after execution of the inference engine program.

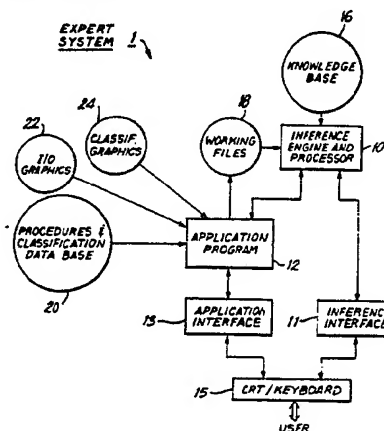


FIG. 1

EXPERT SYSTEM

FIELD OF THE INVENTION

5 The present invention relates to the field of expert systems, and is directed to an expert system intended for use in treating various types of trauma, and in particular, orthopedic trauma.

BACKGROUND OF THE INVENTION

10

Artificial Intelligence (AI) is a branch of science resulting from the marriage of the cognitive and computer sciences. Computers, originally used for the manipulation of numbers (data) are now being used for the manipulation of ideas (knowledge). Trends and solutions can be inferred by the assimilation of
15 observed facts just as numbers are added and subtracted to produce totals. Computer systems are being developed that exhibit thought processes previously ascribed only to humans.

The study of AI leads to insight regarding the human thought processes in addition to the development of practical systems to solve problems in the workplace, the school and the home. The "expert system" is one method of obtaining such practical results with AI.

20 An expert system solves a problem through the manipulation of knowledge. The system consists of an inference engine and a knowledge base. The knowledge base is compiled from the experience of human experts in the field and encoded in a computer language suited for the description of ideas and principles. The inference engine controls the flow of the program, tracing solutions.

The inference engine has, in recent years, become a widely available product through a number of
25 companies, including Gold Hill Computers Inc., of Cambridge, Massachusetts; Intellicorp, of Mountain View, California; Technology Applications, Inc., of Jacksonville, Florida; Teknowledge Inc., of Palo Alto, California; Neuron Data Inc., of Palo Alto, California; and Texas Instruments, Inc., of Austin, Texas. Two inference engines have been disclosed in U.S. patent 4,658,370 to Erman et al., and 4,648,044 to Hardy et al., both assigned to Teknowledge Inc.

30 Expert systems recently have found use in a variety of applications, such as in agriculture, chemistry, computer design, construction, engineering, finance, management, health care, manufacturing, and others. For example, in U.S. patent 4,591,983 to Bennett et al., an expert system for use in inventory control is disclosed, and U.S. patents 4,517,468, 4,642,782, and 4,644,479, all to Kemper et al., each disclose a diagnostic system for monitoring an industrial system, such as a steam turbine generator power plant.

35 In the health care field, hospitals and medical laboratories have used computers to analyze blood and run certain tests. Data bases have been established for recommending drug therapies for certain types of cancers. An expert system made by Cardinal Systems Inc., Minneapolis, Minnesota, includes standard textbook data, and a graphical illustration of the sympathetic nervous system, for purposes of testing a diagnosis, and recommending therapeutic drugs. Other expert diagnostic and treatment systems are
40 specific to a particular healthcare concern, such as, for example, a system called "Senex", specifically designed to aid in the treatment of breast cancer, and a system called "Hepatitis Assistant", designed for better diagnosis and treatment of hepatitis patients. Other health care systems are known to address the specific fields of epilepsy, poison control, childbirth and physical rehabilitation.

45 Although prior art expert systems have been designed to address a relatively wide range of health care concerns, little is known to have been done in the area of treatment of physical trauma. That is, it is believed that none of the existing expert systems designed for health care applications have provided the ability to perform a consultation to help determine the optimal manner in which to treat a specific type of trauma. Such a system would be useful not only for suggesting a treatment, but also for providing a consultation session between an experienced surgeon and a learning surgeon.

50

OBJECTS AND SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide an expert system directed to the treatment of
 5 physical trauma.

It is a further object of the present invention to provide an expert system for the specific field of orthopedic trauma.

It is a further object of the present invention to provide an expert system which provides one or more treatment recommendations based upon specific classifications of physical trauma.

10 It is a further object of the present invention to provide an expert system capable of providing a treatment recommendation based upon specific classes of orthopedic trauma.

These objects are achieved by placing textbook information, such as fracture classification, in a database, and expert information concerning orthopedic fractures in a knowledge base. In use, a fracture to be treated is classified, additional trauma information is obtained and some patient history. Initial treatment
 15 suggestions based upon the classification of the fracture are judged for appropriateness based upon supplemental clinical information, namely the expert information in the knowledge base. During inferencing, additional information may be requested by the computer as needed. Treatment suggestions are presented in the order of preferred use.

The expert system in accordance with the present invention provides the user with one or more
 20 suggested treatments for a patient with physical trauma. The system includes a computing device having a memory, a plurality of databases in the memory, an application program and an inference engine program. The databases include graphical illustrations of different types of physical trauma and the knowledge base which contains treatment information. The application program is executed in the computing device and
 25 interactively displays a series of screens, including at least some of the graphical illustrations, to elicit responses from the user concerning the specific type of physical trauma and specific characteristics of the patient. The inference engine program, which is also executed in the computing device, uses the knowledge base and information related to the responses elicited from the user, for selecting one or more suggested treatments. The application program presents the suggested treatments to the user after the execution of the inference engine program.

30 In accordance with a more specific aspect of the present invention, the physical trauma consists of orthopedic fractures.

BRIEF DESCRIPTION OF THE DRAWINGS

35

These and other objects, aspects and embodiments of the present invention will now be described in more detail with reference to the following drawing figures, of which:

FIG. 1 is a diagram illustrating the structure of the expert system in accordance with the present
 40 invention;

FIGS. 2 through 15 illustrate examples of screens which may be displayed in order to elicit information concerning characteristics of an orthopedic fracture and the patient, from the user of the system;

FIG. 16 illustrates a sample screen which can be used to illustrate to the user of the system the
 45 suggested treatments for the particular orthopedic fracture and patient;

FIG. 17 illustrates a screen which can be used to illustrate to the user of the system possible
 50 complications associated with the particular fracture under study;

FIG. 18 illustrates the hierarchy of information within the knowledge base of the expert system;

FIG. 19 is a chart illustrating the forward and backward chaining employed by the inference engine,
 in accordance with the present invention; and

FIG. 20 is an illustration of a technique for determining the optimal treatment, based on certainty factors and the hierarchy of treatment suggestions.

55

DETAILED DESCRIPTION OF THE INVENTION

The expert system in accordance with the present invention will be described with initial reference to FIG. 1. The expert system 1, includes an inference engine and processor 10, inference engine 11, application program 12 and application interface 13. The inference engine and processor 10 functions as an inference engine, under the control of an inference engine program, and also executes the application program 12, when necessary to perform application program functions, under the overall control of the inference engine. Communication between the expert system and the user is by way of a CRT/keyboard 15 and through the inference interface 11, for communicating with the inference engine and processor 10, and by way of application interface 13, for communicating with the application program 12. The inference engine and processor 10 can be selected from one of the available expert system packages identified above, for example, the Texas Instruments Personal Consultant™ Plus. An advantage of such systems is their ability to run on the commonly available 80286 or 80386 DOS-based personal computers.

The inference engine and processor 10 receives information from two data bases, namely a knowledge base 16, and a data base of working files 18 which are generated by the application program 12, based on information elicited from the user.

The knowledge base 16 includes a collection of rules and parameter descriptions, upon which one or more preferred treatment techniques are based. The information within the knowledge base is based on information from experts within the relevant field, in this case physical trauma, and in particular, orthopedic fractures. The working files 18, comprised of files referred to as PATIENT, DISEASE, TRAUMA and TISSUE, reflect specific information about a patient, including, for example, the patient's specific trauma, characteristics of the surrounding tissue, patient information, such as height, weight, and the like, and information as to any pre-existing conditions, such as osteoporosis.

The knowledge base 16 permanently exists within the expert system, although it should be updated periodically in accordance with currently available expert knowledge. The working files 18, on the other hand, are specific to each patient and injury, and therefore must be created with each use of the expert system. In accordance with the present invention, a procedures and classification data base 20, input-output graphics 22 and classification graphics 24, are provided for the purpose of gathering the requisite patient and trauma information from the system user, and assembling that information into the working files 18. The procedures and classification description data base 20 includes, for each classification of trauma, such as orthopedic fractures, one or more initially recommended treatments and a short description of the orthopedic fracture. The initial procedures are based on textbook, and perhaps expert information, and are used by the inference engine as a starting point from which to determine a final group of recommended procedures. Also included in the data base 20 is a hierarchy of treatment procedures, a compilation of characteristics of each class of fracture, and a compilation of characteristics associated with pre-existing diseases and other trauma that the patient may exhibit. These files will be described in more detail below.

The input-output graphics 22 are provided as part of the expert system to elicit responses to a series of questions relating to the patient and trauma. The classification graphics 24, are based on the particular classes of trauma which the expert system addresses. In accordance with the present example, namely that of orthopedic fractures, the classification graphics 24 are based upon specific types of orthopedic fractures assembled from textbook and similar data. Two such sources of information are M.E. Müller, M. Allgower, R. Schneider and H. Willenegger, *MANUAL OF INTERNAL FIXATION* (Berlin, Heidelberg, New York: Springer-Verlag, 2nd ed. 1979), and M.E. Müller, S. Nazarian and P. Koch, *CLASSIFICATION AO DES FRACTURES, TOME I: LES OS LONGS* (Berlin, Heidelberg, New York: Springer-Verlag, August 1987).

The input-output graphics 22 and classification graphics 24, form the screens illustrated in FIGS. 2-17. It will be appreciated that these screens are exemplary and that other screens, which elicit the proper information, may be equally suitable. Also, the order in which the screens are displayed may be changed as necessary. Finally, although the subject matter of the screens relates to orthopedic fractures, the same or similar techniques may be used to gather information as to other types of trauma as well.

The application program, the knowledge base and the procedures and classification data base, produced in accordance with the specific example of the present invention described herein, are set forth in the appendix.

The operation of the expert system in accordance with the present invention will now be described with reference to FIGS. 2-17. Initially, it will be assumed that the expert system has begun to execute the application program, in a manner which will be described in more detail below.

FIG. 2 is an illustration of the first of the screens provided by I/O graphics 22. As shown in FIG. 2, the expert system is called ORTHOPEDIC ADVISOR. The screen of FIG. 2 provides a "menu", namely a view

of the primary bones in the human body, from which the user can select the particular bone which has been fractured. This can be done through the use of any well-known input technique, such as a touch-screen input, or through the movement of a cursor (not shown) by way of a mouse or cursor keys. The application program, provided in the appendix, supports the use of touch-screen technology and cursor movement through a mouse input or cursor keys. The user will either touch the screen at the location corresponding to the bone under study or move the cursor to that bone, in order to input that information into the application program. In the present example, the selected bone will be the upper long bone of the leg, namely the femur.

Also provided as part of the screen shown in FIG. 2 are location on the screen, designated by reference numerals 28, 30 and 32, commonly called "icons", which may be selected by the user in order to perform the stated functions. For example, icon 28 which bears the legend "last", can be selected by the user either by touching the icon or by positioning the cursor over the icon, in order to instruct the application program to display the previous screen. Similarly, icon 30, which bears the legend "next", may be selected by the user to select the next screen, and icon 32, which bears the legend "stop", may be selected to terminate the program at this point. Each of these icons appears in the screens shown in FIGS. 2 through 16, and since their function is the same, further reference will not be made thereto.

The femur having been selected, the application program displays the screen illustrated in FIG. 3, either automatically upon selection of the particular bone, or in response to selection of icon 30 for the next screen. The screen illustrated in FIG. 3 shows the selected bone, namely the femur, in isolation, and requests the user to select the portion of the femur which has been fractured. This can be accomplished by having the user either touch or move the cursor to the effected portion of the femur 34, or by having the user touch or move the cursor to any one of three areas 36, 38 or 40, to select the proximal femur, the femoral shaft, or the distal femur, respectively. The two-digit number illustrated in the upper left-hand corner of the areas 36, 38 and 40 designate the numerical classification of the site of the fracture, according to a coding scheme used by the procedures and classification data base 20.

After having selected the affected portion of the relevant bone, in this example the proximal femur, the screen illustrated in FIG. 4 is displayed, and the user is requested to select the affected region of the proximal femur, namely the (i) trochanteric region, (ii) the femoral neck, or (iii) the femoral head, classified as locations 31a, 31b and 31c, respectively. As shown in FIG. 4, three illustrations 42, 44 and 46 of the proximal femur are illustrated, each with cross hatching on the respective region, namely the trochanteric region in illustration 42, the femoral neck in illustration 44 and the femoral head in illustration 46. In this manner, the user continues as before, by simply selecting the illustration that corresponds to the fracture under study. In accordance with the present example, the femoral neck is selected as the region of fracture, and in response to that selection, the next screen, illustrated in FIG. 5, is displayed.

FIG. 5 illustrates three possible types of fractures of the femoral neck, namely the femoral neck in abduction (classification 31B1), the femoral neck with a vertical fracture line (classification 31B2), and the femoral neck in adduction (classification 31B3). These classifications of specific fracture types are taken from the relevant body of knowledge on the subject of orthopedic fractures, such as the treatises by M.E. Müller et al., referred to above. As before, three illustrations 48, 50 and 52 are provided for the respective fracture classifications, in order to assist the user in selecting the appropriate classification. The user selects the type of fracture experienced by the patient, which in this example is the femoral neck with vertical fracture line (classification 31B2). In response, the screen illustrated in FIG. 6 is displayed, illustrating three different types of femoral neck fractures with vertical fracture lines, specifically, a fracture of the basilar neck (classification 31B2.1), the medial neck (classification 31B2.2) and a subcapital fracture (classification 31B2.3). Three illustrations 54, 56 and 58 are provided in order to assist the user in selecting the appropriate type of fracture. In the present example, the medial neck (classification 31B2.2) is selected, and in response, the screen illustrated in FIG. 7 is displayed.

The screen illustrated in FIG. 7, which is generated in accordance with the graphics information in I/O graphics 22, requests that the user provide information as to any other trauma that the patient has experienced. For example, if the patient has lost blood, the user would select icon 60 to indicate cardiovascular trauma. In response to this selection, the screen illustrated in FIG. 8 is displayed and the user indicates the severity of blood loss, by selecting one of the choices 62. After making such a selection, the screen illustrated in FIG. 7 is again displayed, and a legend indicating the severity of cardiovascular trauma will appear next to icon 60. The user can then select other types of trauma, if applicable. This information is placed into the file TISSUE, in the working files 18, and is also used by the application program to arrive at an injury severity score (ISS), which is similarly placed in the file TISSUE. The calculation of the ISS, is based on the technique disclosed in the article by the American College of Surgeons Committee on Trauma, entitled "Field Categorization of Trauma Patients and Hospital Trauma

Index", BULLETIN OF THE AMERICAN COLLEGE OF SURGEONS, Vol. 65, February 1980, pp. 28-33.

The next screen displayed is illustrated in FIG. 9 in which the user specifies the damage to the soft tissue surrounding the fracture and the open grade. This information also goes to the file TISSUE.

The screen illustrated in FIG. 10 is then displayed and the user indicates the patient's weight, age, height and sex. In this example, the patient weighs 60 kilograms, is 35 years old, is 170 centimeters tall and is male. This screen readily lends itself to touch-screen applications, but can also be used with cursor movement as well. The next screen, showing in FIG. 11, requests the user to input information concerning the patient's occupation or lifestyle, in the context of the injury, i.e., whether the patient is sedentary, active or vigorous. In this example, the patient is considered "active". The information elicited from the user by the screens shown in FIGS. 10 and 11 is placed in the PATIENT file in the working file 18.

The next screen, shown in FIG. 12, requests the user to indicate any pre-existing illnesses that the patient might have, or other treatment concerns about the patient, since such considerations could affect the patient's ability to tolerate surgery, to heal properly, to remain convalescent (recumbency), or to follow instructions, for example. Six icons 64 designate selections of the following treatment concerns: surgery, healing, recumbency, patient reliability, bone quality and specific diseases. The first five of these treatment concerns are used by the application program to enter information, if applicable, into the file DISEASE in the working files 18. The sixth health concern, namely SPECIFIC DISEASES, elicits information from the user to determine whether any of the patient's pre-existing diseases would cause one of the first five health concerns, and that information would be placed into the file DISEASE, as well.

Specifically, when the user selects SPECIFIC DISEASES as a treatment concern, the application program displays the screen illustrated in FIG. 13 and requests that the user indicate, using icons 66, the physiological system associated with the patient's specific illness. In this example, the cardiovascular system is selected, and in response to that selection, a menu 68, FIG. 14, is displayed, and the user indicates one of the particular types of specific cardiovascular diseases. In this case, vascular insufficiency is selected. In response to this selection, the application program inquires of the procedures and classification data base 20 to see what specific treatment concerns are caused by vascular insufficiency. Using the DISEASE data base in the procedures and classification data base 20 (a copy of which is provided in the appendix), it is determined that vascular insufficiency causes a surgery concern with a 50 percent certainty factor. Thus, in response to selecting vascular insufficiency, the application program displays the screen illustrated in FIG. 15, to inform the user of the surgery concern. Alternatively, any of the surgery, healing, recumbency, reliability and bone quality concerns can be input directly through the use of the first five icons 54, FIG. 12.

After entering the information concerning treatment concerns, the user can now instruct the expert system to proceed with the consultation, by selecting the NEXT icon in FIG. 15. In response, the inference engine 14, FIG. 1, applies the rules of the knowledge base 16, to the information contained within the working files 20 concerning the specifics of the patient and the orthopedic fracture. In the event that further information is required, for a particular set of inputs, the inference engine may generate one or more further inquiries through the expert system interface 11, in order to gather the additional data from the user. For example, the expert system may inquire as to whether the injury resulted from a simple fall, from which the inference engine might infer that the patient is osteoporotic, if such information is considered to be important.

When the inference engine has gathered all the necessary information, it completes its tasks by achieving certain "goals", and then returns temporary control to the application program. A screen is then displayed, as illustrated in FIG. 16, and includes a table 70 which shows the classification of the fracture, and selected characteristics of the patient and trauma. In addition, four choices 72, 74, 76 and 78 are displayed as the treatments selected by the inference engine, for this specific case. These treatments may be ordered in terms of the most to least highly suggested, either by positional order, as displayed, or by a numerical or alphabetic indication by each treatment. In this case, the suggested procedures for this patient are an A-frame, a hip prosthesis, a DHSTM implant and lag screws. Additionally, the user at this point, by selecting a particular one of the recommended treatments, for example the DHSTM implant (the treatment indicated by the reference numeral 76), the application program will actually show the DHSTM implant 80 in place.

Finally, when the user is finished with the screen shown in FIG. 16, a further screen, illustrated in FIG. 17, can be displayed to illustrate to the user a statistical summary of complications experienced with patients with similar fractures, if such a data base is available.

The program flow will now be described with reference to an example which uses an inference engine

developed by Texas Instruments, called Personal Consultant™ Plus, Version 2.0. It will be appreciated, however, that other inference engines, such as those associated with some of the above-mentioned commercially available expert systems could be used as well. Reference will also be made to the following programs and data bases which are attached hereto as an appendix:

5

10

15

Program or Data Base	Appendix
Application Program	Pages 1-70
Knowledge Base Rules and Parameter Descriptions	Pages 71-93
Procedure Hierarchy	Page 94
Initial Procedure Suggestions	Pages 95-96 (97 not used)
Classification Descriptions	Pages 98-102
Batch Files	Pages 103-105
Preexisting Diseases	Page 106
Trauma Descriptions	Page 107
Classification Expansions	Pages 108-109

The application program, at pages 1 through 70 of the appendix, generally corresponds to the application program 12 of FIG. 1. It is written in MICROSOFT C programming language for use on IBM AT compatible computers, and makes use of a data base program called BTRIEVE, by SoftCraft, of Austin, Texas, and of a graphics program called ESSENTIAL GRAPHICS, Version 1.5, by Essential Software, Maplewood, New Jersey.

The Knowledge Base Rules and Parameter Descriptions at pages 71 through 93 of the appendix, form the knowledge base 16 of FIG. 1. The Procedure Hierarchy, Initial Procedure Suggestions, Classifications Descriptions, Preexisting Diseases, Trauma Descriptions and Classification Expansion databases form the database 20 of FIG. 1.

To initiate the program the user will type the command "AOPC", which will call the batch file "AOPC.BAT" found in the batch file listings in the appendix. That batch file in turn causes the inference engine program to be executed, to thereby create the inference engine, which controls all further program flow. The inference engine first requests the name of the knowledge base with which it is to work, and by responding "AOOA", the name given to the knowledge base of the present example, the inference engine accesses the appropriate knowledge base.

The inference engine first examines a listing of parameters entitled FRAMETYPES in the knowledge base, in order to determine the structure of the knowledge base. The parameter group FRAMETYPES is set forth on page 84 of the appendix, but the structure it implies is also shown in FIG. 18.

With reference to that figure, the rules and parameters of the knowledge base, and associated "goals", are divided into logical categories, commonly referred to as "frames". In the present example, four such frames exist, namely, ADVISE, PROS-CONS, PATIENT and TRAUMA. The parameter group FRAMETYPES indicates to the inference engine the hierarchy of frames, as shown in FIG. 18. The first, and highest level frame is ADVISE, followed by PROS-CONS, which in turn is followed by the PATIENT and TRAUMA frames, which share the lowest level in the hierarchy. Also shown in FIG. 18 for the ADVISE and PROS-CONS frames are goal associated with each frame. No goals are assigned to the PATIENT and TRAUMA frames.

The rules within each frame are logically grouped according to function. For example, the PATIENT frame contains rules for determining specific characteristics of the patient, for example whether the patient is young or old, whether the patient is osteoporotic, and the like. The rules in the TRAUMA frame determine characteristics of the trauma, for example whether the trauma involves bilateral fractures of weight-bearing bones. The rules within the PROS-CONS frame rely upon the inferences drawn, and the conclusions reached, in the PATIENT and TRAUMA frames. For example, if it is determined from the PATIENT frame that the bone site is osteoporotic, then the rules in the PROS-CONS frame might determine that there is strongly suggestive evidence that a procedure which allows for settling to occur is important. Finally, the rules within the ADVISE frame rely upon the inferences and conclusions drawn in the PROS-CONS, PATIENT and TRAUMA frames, and based upon the inferences and conclusions drawn in those frames, the ADVISE frame arrives at a final set of recommended procedures.

The inference engine will inspect the list of frametypes and determine from that list the highest order frame, which in this example is ADVISE. The inference engine then looks at the goals stated for the ADVISE frame. As shown in FIG. 18, the stated goals are INPUT-READ, TREATMENT, and FINISHED. The inference engine, upon instantiating any of the frames shown in FIG. 18, including the ADVISE frame, tries

to achieve the stated goals, in the stated order. In the case of the ADVISE frame, the first such goal is INPUT-READ. The inference engine then searches for the parameter "INPUT-READ" within the parameter descriptions in the knowledge base, and the listing as it appears in the knowledge base (and appendix) is set forth immediately below, to help facilitate this explanation:

5

INPUT-READ [ADVISE-PARMS]

10 TRANSLATION: (information available from initial system queries)
 TYPE: YES/NO
 USED-BY: RULE026 RULE007 RULE025 RULE038 RULE039
 UPDATED BY: SREFMARK RULE026

The parameter INPUT-READ is indicated as being one of the ADVISE parameters. Its translation, for the
 15 convenience of the system designer, is indicated as taking information from initial system queries. The type of parameter is indicated as being "yes/no", i.e. a single valued variable, as opposed to a variable which can take on numerous values. The INPUT-READ parameter is indicated as being used by Rules 26, 7, 25, 38 and 39, and updated by Rule 26, which is further indicated as being a self-referencing rule, meaning that the parameter "INPUT-READ" is found in both the "if" and "then" portions of the rule, as explained in more
 20 detail below.

The inference engine then branches to the rule that updates INPUT-READ, namely Rule 26, which executes a disk operating system call to a batch file INITIT.BAT. This batch file, which also appears in the appendix, calls the application program 12, FIG. 1.

Execution of the application program creates the working files 18, FIG. 1, namely TRAUMA, TISSUE,
 25 PATIENT and DISEASE, in response to the information elicited from the user, in combination with the information contained in the databases in database 20. Specifically, the TRAUMA file includes the following parameters:

FRACTURE CLASS
 ARTICULAR
 30 INHERENTLY STABLE
 NUMBER-PIECES
 POTENTIALLY STABLE
 WEIGHT-BEARING

The application program designates the FRACTURE CLASS in the TRAUMA file to be the class of
 35 fracture designated by the user in the screen illustrated in FIG. 6, namely classification 31B2.2. The remaining parameters in the TRAUMA file are given values based upon the CLASS EXPANSION database, in database 20, which, for each fracture classification provides information as to whether the fracture is articular, inherently stable, etc. It will also be noted that one or more initially indicated treatments are provided by the Initial Procedure Suggestion data base, based upon the fracture classification.

40 The file TISSUE includes the following parameters:

EXTREMITY
 NERVOUS
 RESPERATORY
 ABDOMINAL
 45 CARDIOVASCULAR
 SKIN
 COMPLICATIONS
 ISS
 SOFT TISSUE DAMAGE
 50 OPEN CLASS

The first seven parameters of the TISSUE file are designated by the application program in response to the information provided by the user in response to the screen illustrated in FIG. 7. The SOFT TISSUE DAMAGE and OPEN CLASS parameters are designated in accordance with the information provided by the user in response to the screen illustrated in FIG. 9. Finally, the injury severity score (ISS) is calculated by
 55 the application program, based on the other parameters in the TISSUE file.

The PATIENT file contains the following parameters, each of which is taken directly from the information given by the user in response to the screens illustrated in FIGS. 10 and 11:

AGE

SEX
WEIGHT
HEIGHT
LIFESTYLE/OCCUPATION

5 The DISEASE file contains the following parameters:

HEALING
OSTEOPOROTIC
CONDOLESCENT RISK
SURGERY INTOLERABLE

10 UNRELIABLE

These parameters are either taken directly from the user inputs in response to the screen illustrated in FIG. 12, or through use of the Preexisting Disease database, in database 20, if the user selects one or more of the diseases set forth in the screen illustrated in FIG. 13.

15 Upon completion of the creation of the working files 18, the application program relinquishes control to the inference engine, which, while still in the ADVISE frame, checks off the INPUT-READ goal of that frame as being completed. The inference engine and processor 10 finds the next goal, TREATMENT, in the ADVISE frame, and searches the knowledge base 16 for the parameter TREATMENT, which is reproduced below:

20

TREATMENT [ADVISE-PARMS]

TRANSLATION: (the suggested treatment for this orthopedic fracture)

25 LEGALVALUES: TEXT

TYPE: MULTIVALUED

UPDATED-BY: RULE010 RULE005 RULE009 RULE004 RULE002 RULE003 RULE032 RULE008 RULE007

30 The goal TREATMENT is to determine one or more preferred treatments for the specific patient and trauma. The above listing indicates that the TREATMENT parameter is multivalued and is updated by Rules 10, 5, 9, 4, 2, 3, 32, 8 and 7. The inference engine selects the first rule which updates the TREATMENT parameter, namely Rule 10, and searches the knowledge base for that rule. It should be noted that all rules in the knowledge base must be in either the same frame as the parameter it updates (TREATMENT in this case), or in a lower frame. Rule 10, which is found in the ADVISE frame, is set forth below:

35

RULE 010 [ADVISE RULES]

IF: VERSATILE

40 THEN: TREATMENT = "X-FIX" CF 3 AND TREATMENT = "CAST" CF 7 AND TREATMENT = "TRACTION" CF 7 AND TREATMENT != "PLATE" CF 1 AND TREATMENT != "NAIL" CF 3 AND TREATMENT = "LAG SCREWS" CF 3 AND TREATMENT = "WIRE" CF 3 AND TREATMENT = "OPEN REDUCTION" CF 3

45 An interpretation of the above rule is as follows. If a treatment that is versatile in its application is important, then there is weakly suggestive evidence (with a certainty factor of 3%) that the suggested treatment for this orthopedic fracture is X-FIX, and there is weakly suggestive evidence (certainty factor of 7%) that the suggested treatment for this orthopedic fracture is CAST, and so on. An exclamation point before an equal sign indicates that the stated treatment is not recommended, with a given certainty factor. For example, Rule 10 states that there is weakly suggestive evidence (certainty factor of 1%) that the suggested treatment is not PLATE.

50 The inference engine now tries to determine the truth of the premise of this Rule, namely whether a treatment that is versatile in its application is important. Thus, the parameter VERSATILE becomes the inference engine's new goal. The inference engine 14 searches the knowledge base 16 for the parameter VERSATILE, and that parameter (found in the ADVISE frame) is reproduced below:

55

VERSATILE [ADVISE PARMS]

TRANSLATION: (a treatment that's versatile in its application is important)

DEFAULT: (NO)

5 TYPE: YES/NO

USED-BY: RULE010

UPDATED-BY: RULE041

10 If a value of VERSATILE cannot, for one reason or another, be calculated, the default value for the parameter is "No". VERSATILE is defined as a single valued parameter (Yes/No), is used by Rule 10 and is updated by Rule 41. The inference engine will search the knowledge base for Rule 41 which is reproduced below:

15 RULE 041 [TRAUMA-RULES]

IF: ODD SIZE OR VALUE ISS > 30

THEN: VERSATILE CF 60

20 The inference engine will at this point note that Rule 41 is located within the TRAUMA frame, FIG. 18, rather than in the ADVISE frame, where the parameter VERSATILE is located. At every instance, or "instantiation" in which the inference engine must move from one frame to another, it may only do so through a contiguous frame in the hierarchy, shown in FIG. 18. Thus, in order for the inference engine to move from the ADVISE frame to the TRAUMA frame, it must first instantiate the PROS-CONS frame.

25 Upon instantiating the PROS-CONS frame, the inference engine selects all of the goals within the PROS-CONS frame as its current set of goals, with the goals being taken in the order listed in the FRAMETYPE file, as shown in FIG. 18. Thus, the parameter STRONGER becomes the next goal, or more properly "sub-goal" of the inference engine.

30 The inference engine searches the knowledge base for the parameter STRONGER, in the current frame (PROS-CONS) or a higher frame (ADVISE). The STRONGER parameter is found in the ADVISE frame and is listed below:

STRONGER [ADVISE-PARMS]

35 TRANSLATION: (a relatively very strong system is important)

DEFAULT: (NO)

TYPE: YES/NO

USED-BY: RULE002

40 UPDATED-BY: RULE011

Since this parameter is updated by Rule 11, the inference engine searches for that rule in the current (ADVISE) or lower frames. Rule 11 will be found in the PROS-CONS frame, and is listed below:

45 RULE 011 [PROSCONS-RULES]

IF: BI-WEIGHT

THEN: STRONGER CF 90

50 The rule states that if the trauma involves bilateral fractures of weight-bearing bones, there is strongly suggestive evidence (90%) that a relatively strong system is important. The inference engine must now find the parameter BI-WEIGHT in the knowledge base, and the parameter is defined therein as follows:

55

BI-WEIGHT [PROSCONS-PARMS]

TRANSLATION: (the trauma involves bilateral fractures of weight bearing bones)

PROMPT: YES

5 TYPE: YES/NO

USED-BY: RULE040

The designation "PROMPT: YES" indicates that the inference engine will prompt the user to provide or confirm this information, if necessary. Since the parameter BI-WEIGHT is updated by Rule 40, the inference engine searches the knowledge base for Rule 40, which is found in the TRAUMA frame, as follows:

10

RULE 040 [TRAUMA-RULES]

15 IF: VALUE EXTR < 4

THEN: ! BI-WEIGHT

This rule states that if the extremity value is less than four, then the trauma does not involve bilateral fractures of weight-bearing bones. The inference engine searches the knowledge base for the parameter EXTR, and it is found in the TRAUMA frame, as follows:

20

EXTR [TRAUMA-PARMS]

25 TRANSLATION: (the extremity score from the health trauma index)

TYPE: SINGLEVALUED

USED-BY: RULE040

UPDATED BY: RULE039

Since this parameter is updated by Rule 39, the inference engine searches the knowledge base for Rule 39, which is located in the TRAUMA frame, as follows:

30

RULE 039 [TRAUMA-RULES]

35

IF: INPUT-READ

THEN: READ*FRAME "tissue" QUOTE (EXTR NERV RESP ABDM CARD SKIN COMP ISS SOFT-TISSUE-DAMAGE OPEN-CLASS) TALLY

This rule states that if information is available from the initial system queries, then the stated data are to be read from the file TISSUE. Since the premise is true, namely that information is available from the initial system queries, then the inference engine reads the stated values from the file TISSUE in the working files 20, FIG. 1, including the value of the extremity score EXTR. Having determined the value of the extremity score, the inference engine is now ready to arrive at a value for the sub-goal STRONGER. This is accomplished through a process called "backward-chaining" and involves stepping back through the path that resulted in the value for the parameter EXTR. With reference to FIG. 19, it will be seen that the value of EXTR was determined by invoking Rule 11, then Rule 40, and then Rule 39, through a "forward chaining" process. The backward chaining takes these rules in the reverse order: Rule 39, then Rule 40, then Rule 11. From Rule 39, Rule 40 is invoked, namely that if the extremity value is less than four, then the trauma does not involve bilateral fractures of weight bearing bones. Since the value of the extremity score is three, then the rule is satisfied, and the trauma does not involve bilateral fractures of weight-bearing bones. The next rule in the chain, Rule 11, is tested, and since the premise of bilateral fractures of weight-bearing bones is not true, then Rule 11 fails and no action is taken. At this point, the inference engine will return to the parameter definition of STRONGER to see whether it is updated by any rules in addition to Rule 11, and if so, tests those rules in the same manner, to see whether they assign a value to the parameter. Since there are no other rules which update the parameter STRONGER, the inference engine selects for the parameter STRONGER the default value, in this case "NO". In essence, the inference engine has just concluded that a relatively strong system is not important.

55

It should be noted that for single-valued parameters, such as STRONGER, which require an answer in

the form of a YES/NO, MALE/FEMALE or YOUNG/OLD, the inference engine will try all of the rules which update that parameter until the rule tests true, and no further rules which update the parameter will be tested. However, for multiply valued parameters, the inference engine will repeat the process for all of the rules which update that parameter, until all such rules are exhausted. In the present example, the only
 5 multiply valued parameter is TREATMENT -- all others are single valued.

Returning to FIG. 18, it will be seen that the first listed sub-goal in the PROS-CONS frame, namely STRONGER, has been determined, and the inference engine will proceed with the remaining sub-goals in the order listed. However, for the purposes of illustrating the present example, it will now be assumed that all of the sub-goals preceding VERSATILE have been satisfied, and the explanation will proceed with the
 10 sub-goal VERSATILE.

The inference engine searches the knowledge base for the parameter VERSATILE and its parameter description, stated above, indicates that VERSATILE is updated by Rule 41, also stated above. Rule 41 states that if the patient is either very large or the injury severity score is greater than 30, there is evidence, having a certainty factor of 60%, that a treatment that is VERSATILE in its application is important. The
 15 inference engine then searches for the parameter ODD-SIZE, and will find the following parameter description in the PROS-CONS frame:

20 ODD-SIZE [PROSCONS-PARMS]

TRANSLATION: (the patient is very large)

PROMPT: YES

TYPE: YES/NO

25 USED-BY: RULE041

UPDATED-BY: RULE021 RULE022

The inference engine will then search for the first rule that updates the parameter, namely Rule 21, which is found in the PATIENT frame as follows:

30

RULE 021 [PATIENT-RULES]

IF: VALUE HEIGHT > 198 OR VALUE WEIGHT > 136

35 THEN: ODD SIZE

Thus, if the height of the patient is greater than 198 centimeters or the weight of the patient is greater than 136 kilograms, then the patient is considered to be very large. The inference engine then searches the knowledge base for the parameter HEIGHT which is found in the patient frame as follows:

40

HEIGHT [PATIENT-PARMS]

TRANSLATION: (the height of the patient in centimeters)

45 PROMPT: YES

EXPECT: POSITIVE-NUMBER

RANGE: (30-230)

TYPE: SINGLEVALUED

USED-BY: RULE021 RULE022

50 UPDATED-BY: RULE 025

The inference engine then searches for Rule 25 which is found in the patient frame, as follows:

55

RULE 025 [PATIENT-RULES]

IF: INPUT-READ

THEN: READ*FRAME "patient" QUOTE (AGE SEX WEIGHT HEIGHT LIFE OCC) TALLY

5 The inference engine, in response to reading Rule 25 will read from the file PATIENT in the working files 20, FIG. 1, the stated parameters, including HEIGHT and WEIGHT.

10 The inference engine will then chain back to Rule 21 (stated above) to see whether the ODD-SIZE premise is true. For this example, it will be assumed that the patient's height is greater than 198 centimeters or that the patient's weight is greater than 136 kilos, and therefore ODD-SIZE is true. Had this not been the case, the inference engine would have returned to the parameter description for ODD-SIZE (stated above) to find that it is also updated by Rule 22, to see whether Rule 22 would yield a true value for ODD-SIZE. However, since Rule 21 yielded a true value, and since the ODD-SIZE parameter is single-valued, the inference engine can ignore Rule 22.

15 Continuing the backward chaining process, the inference engine returns to Rule 41, and since ODD-SIZE is true, then it is also true with a 60% certainty factor that a treatment that is VERSATILE in its application is important.

20 Having determined the value of the sub-goal VERSATILE, the remaining sub-goals are determined in the same manner, and then the inference engine returns to the rule which originally instantiated the PROS-CONS frame, namely Rule 41. However, Rule 41 has already been satisfied, so the inference engine chains back to the previous rule, namely Rule 10.

25 Rule 10 (stated above) starts by indicating that there is a 3% certainty factor that the suggested treatment is an X-FIX, if a VERSATILE treatment is important. In this example, it was determined that a VERSATILE treatment was indicated as being important with a certainty factor of 60%. The certainty factor of 3% stated for the X-FIX treatment in Rule 10 is therefore multiplied by the 60% certainty factor that a VERSATILE treatment is important, and that result is added to a number related to the previous value of X-FIX if any exists, either from a previous calculation of the same type, from a different rule which assigned a value to X-FIX, or from a value assigned to the initial treatments selected by the Initial Procedure Suggestions data base, based on the fracture classification, as described above. The updated total for X-FIX as calculated by Rule 10 is related to these quantities according to well known formulas employed by the commercially available inference engines, or for example, the formulas stated in U.S. patent 4,648,044 to Hardy et al. The remaining values of the parameters CAST, TRACTION, PLATE, and the others set forth in Rule 10 are calculated in the same manner, based on the 60% certainty factor associated with a VERSATILE treatment and any previous value for the particular parameter.

35 After processing Rule 10, the inference engine returns to the parameter which originally invoked it, namely TREATMENT, which is stated above. The next rule which updates the TREATMENT parameter is Rule 5, which is as follows:

40 RULE 005 [HOUSE-RULES]

IF: ACCURATE

THEN: TREATMENT = "X-FIX" CF 13 AND TREATMENT = "CAST" CF 0 AND TREATMENT != "TRACTION" CF 17 AND TREATMENT = "NAIL" CF 23 AND TREATMENT = "LAG SCREWS" CF 30
 45 AND TREATMENT = "WIRE" CF 0 AND TREATMENT != "JOINT REPLACEMENT" CF 17 AND TREATMENT != "OPEN REDUCTION" CF 20 AND TREATMENT = "PLATE" CF 30.

50 The premise of Rule 5 is the need for a relatively accurate result, and the inference engine performs the same chaining functions necessary to determine whether an accurate result is necessary, and with what certainty factor, in a manner similar to that performed for the parameter VERSATILE. After determining a certainty factor for the parameter ACCURATE, the inference engine performs the same mathematical functions with the certainty factors associated with the parameters listed in Rule 5, and with the certainty factor for the parameter ACCURATE, and those values are added to the previous values, if any, for each of the possible treatments.

55 Returning to the description of the parameter TREATMENT, the same procedure is followed for Rules 9, 4, 2, 3, 32, 8 and 7, until final values are generated for each of the possible treatments referenced in those rules. At the conclusion of each of those rules, the inference engine is done with the goal TREATMENT, and the next goal in the ADVISE frame is FINISHED. The inference engine will search the knowledge base for the description of the parameter FINISHED, which functions to call an application

program called FINAL (or FINCUR, if a touch screen is not available). This program presents the debriefing screens illustrated in FIGS. 16 and 17, based on a selected number of the most highly indicated treatments, namely those with the highest values.

In addition to calculating the values for each treatment in accordance with the procedure outlined above, additional processing, in accordance with the present invention, may be made in order to increase the likelihood that the proper treatment or treatments will be selected.

Specifically, each of the recommended treatments forms part of a hierarchy of suggested treatments, as indicated by the file Procedure Hierarchy data base in the working files 20. A chart illustrating the hierarchy for some selected treatments is shown in FIG. 20. The broadest treatment is INTERNAL, and two types of internal treatments are IMPLANT and JOINT REPLACEMENT. Furthermore, two types of implants are PLATE and NAIL. Likewise, two types of plates are DHSTM, and CONDYLAR, and two types of nail treatments are ENDER'S and LOCKING. Finally, under joint replacement, the two treatments are HIP PROSTHESIS and KNEE REPLACEMENT.

The treatment hierarchy shown in FIG. 20 has four levels, numbered 0 through 3, as shown. The broad category of INTERNAL treatments is at level 0, IMPLANT and JOINT REPLACEMENT are at level 1, PLATE, NAIL, HIP PROSTHESIS and KNEE REPLACEMENT area at level 2 and DHSTM, CONDYLAR, ENDER'S and LOCKING are at level 3. A total score for each of the most specific treatments (DHSTM, CONDYLAR, ENDERS, LOCKING, HIP PROSTHESIS and KNEE REPLACEMENT) can be determined as follows. The certainty factors associated with each of the treatments shown in FIG. 20, as determined by the rules implicated by the TREATMENT goal and indicated in parenthesis underneath each treatment, are multiplied by a weighting factor, such that the certainty factors associated with the more specific treatments are more heavily weighted than those associated with the broader treatment categories. In the present example, a weighting factor equal to four plus the number of the level of treatment, equals the weighting factor. As shown the certainty factors are multiplied by 4, in level 0, by 5 in level 1, 6 in level 2, and so on. It should be noted that this specific weighting function is exemplary, and others may be found to be more suitable for particular applications.

After multiplying the certainty factor of each treatment suggestion by the weighting factor, the individual totals for each of the most specific (highest level) treatments are added to the individual totals in each of its parent categories, up to and including the broadest category (level 0). For example, the individual total for DHSTM, is 560. This figure is added to the figure for its parent (PLATE = 90), its grandparent (IMPLANT = 85) and its great-grandparent (INTERNAL = 60), to yield a total score of 795. The higher the total score, the more highly the specific treatment is recommended. A selected number of the most highly recommended treatments may be displayed, as illustrated in FIG. 16. It will be noted that in FIG. 16, the information within icons 66, 68, 70 and 72 generally indicate the treatment hierarchy for the respective treatment.

Thus, the present invention provides a highly sophisticated system for providing a set of recommended treatments for specific categories of physical trauma, using state-of-the-art expert system technology. Various changes and variations to the present invention will occur to those skilled in the art in view of the foregoing description. For example, other types of physical trauma, in addition to orthopedic fractures will find equally suitable implementation using the techniques in accordance with the present invention. It is also intended that the particular classifications of orthopedic fractures, treatments, and other database information be exemplary, rather than limiting, and that all such changes and variations be encompassed so long as the present invention is employed, as defined by the following claims.

Claims

1. An expert system for providing to a user one or more suggested treatments for a patient with physical trauma, comprising:
 - a computing device having a memory;
 - a plurality of data bases in the memory including graphical illustrations of different types of physical trauma and a knowledge base having treatment information;
 - an application program, for execution in the computing device, for interactively displaying a series of screens including at least some of the graphical illustrations, to thereby elicit responses from the user concerning the specific type of physical trauma and specific characteristics of the patient; and
 - an inference engine program, for execution in the computing device, for use with the knowledge base and information related to the responses elicited from the user, for selecting the one or more suggested

treatments;

the application program presenting the suggested treatments to the user after completion of the execution of the inference program.

2. An expert system for providing to a user one or more suggested treatments for a patient with an
 - 5 orthopedic fracture, comprising:
 - a computing device having a memory;
 - a plurality of data bases in the memory, including graphical illustrations of different classifications of orthopedic fractures and a knowledge base having treatment information;
 - an application program, for execution by the computing device, for interactively displaying a series of
 - 10 screens on a display, including at least some of the graphical illustrations, to thereby elicit responses from the user concerning the specific classification of orthopaedic fracture, and specific characteristics of the patient; and
 - an inference engine program, for execution in the computing device, for use with the knowledge base and information related to the responses elicited from the user, for selecting the one or more suggested
 - 15 treatments;
 - the application program presenting the suggested treatments to the user after completion of the execution of the inference program.

20

25

30

35

40

45

50

55

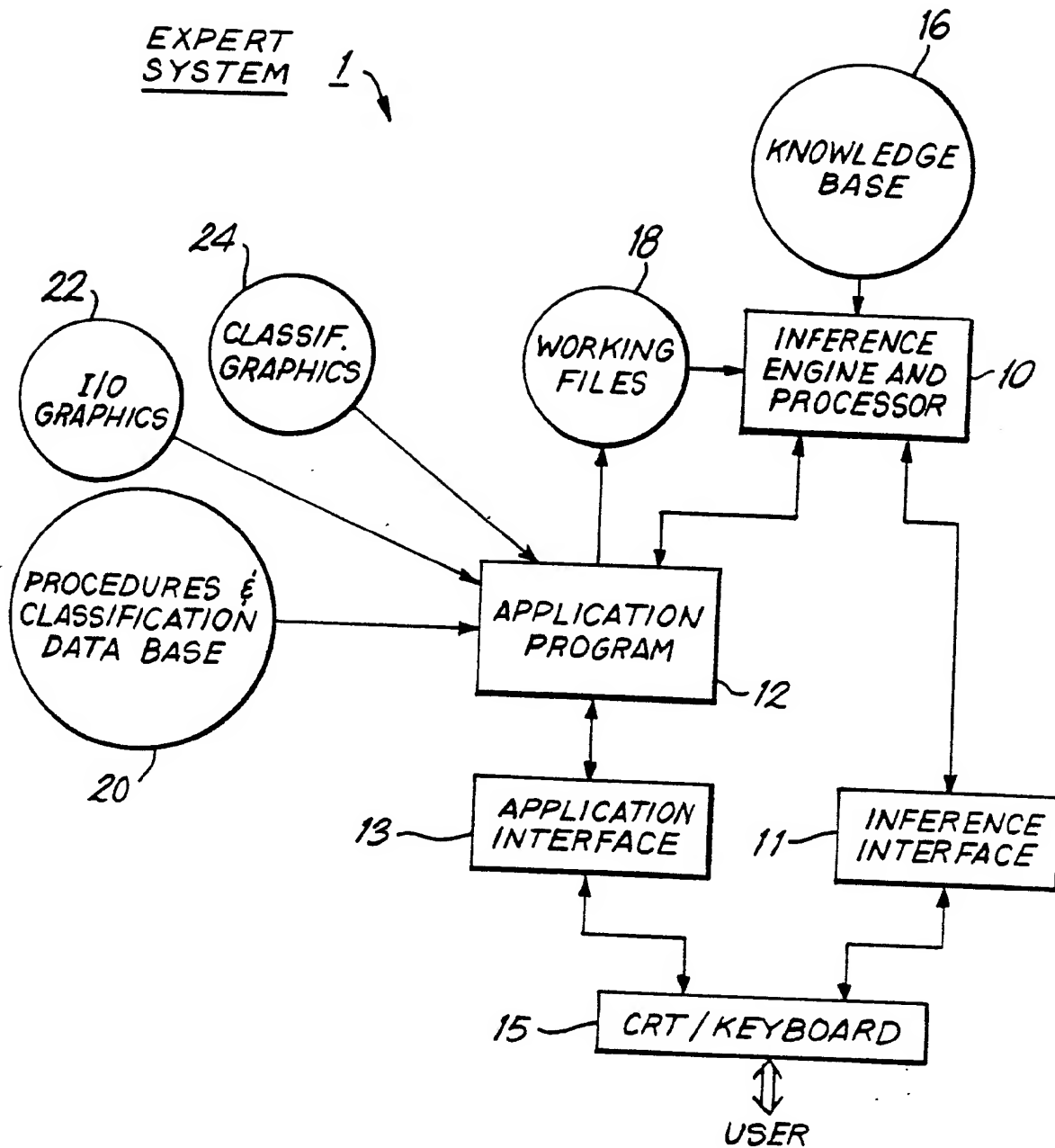


FIG. 1

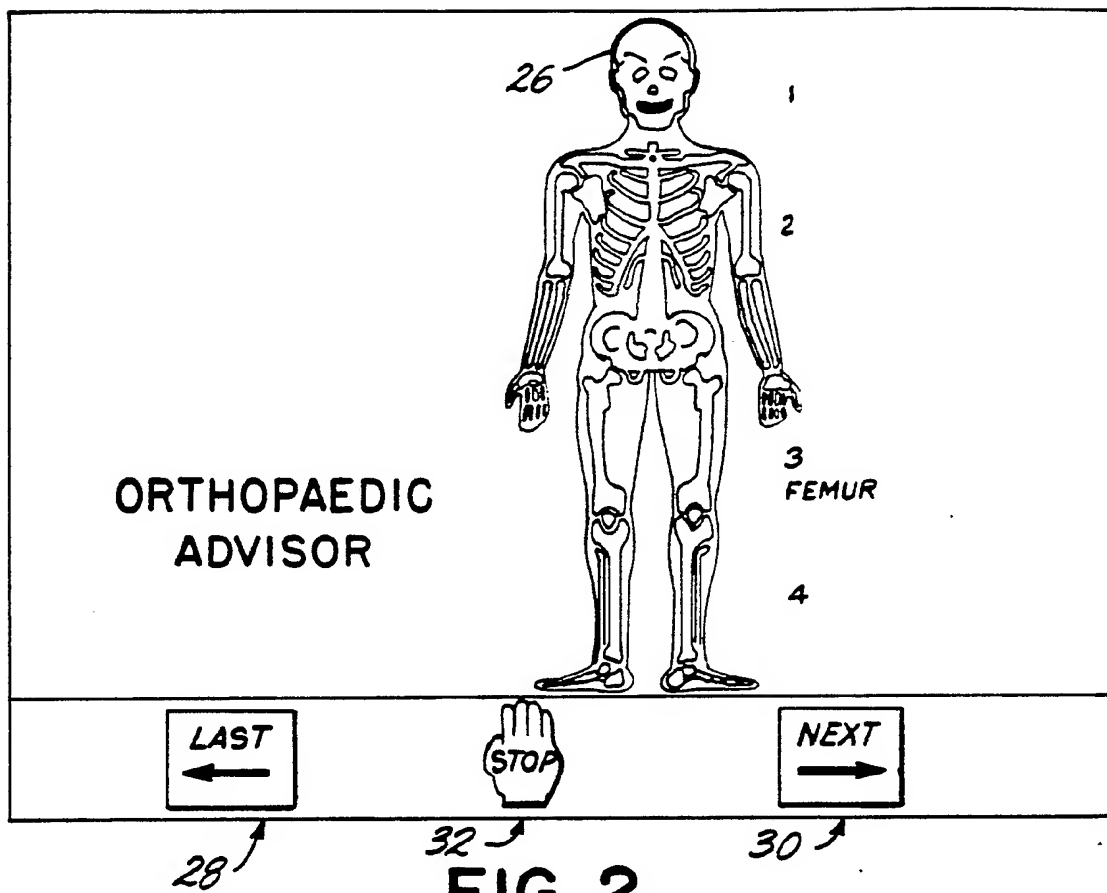


FIG. 2

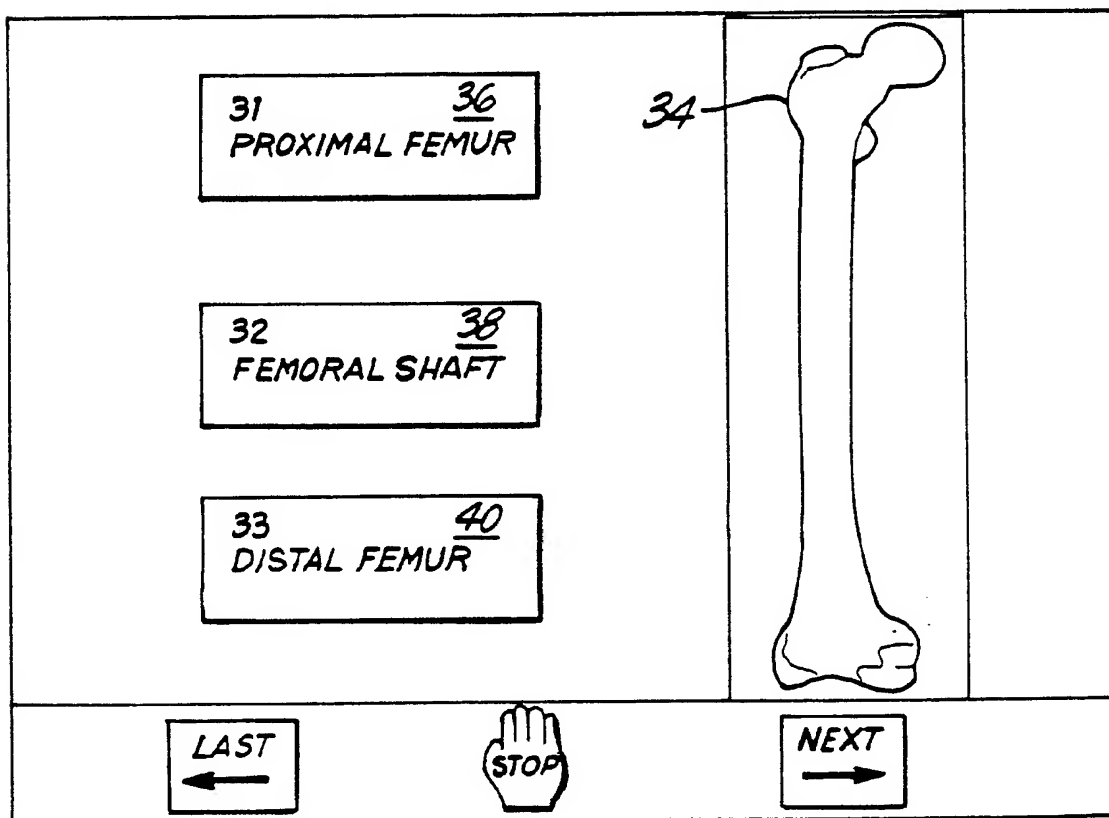


FIG. 3

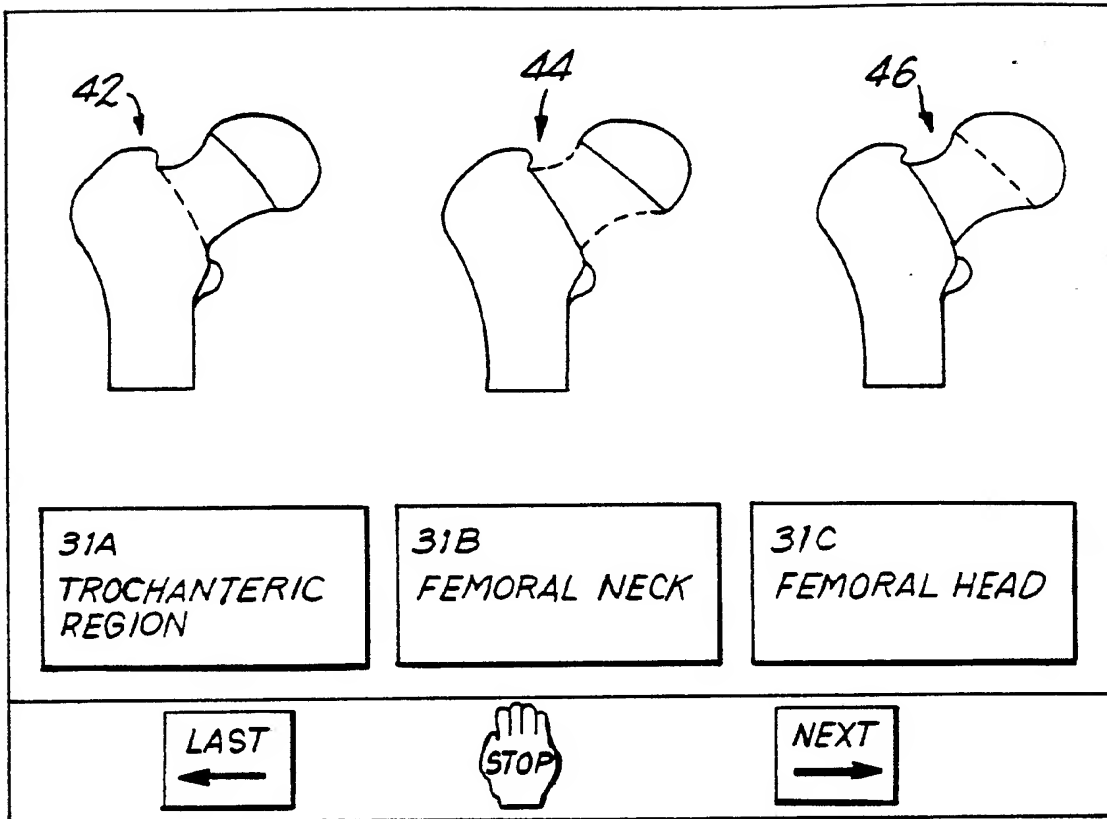


FIG. 4

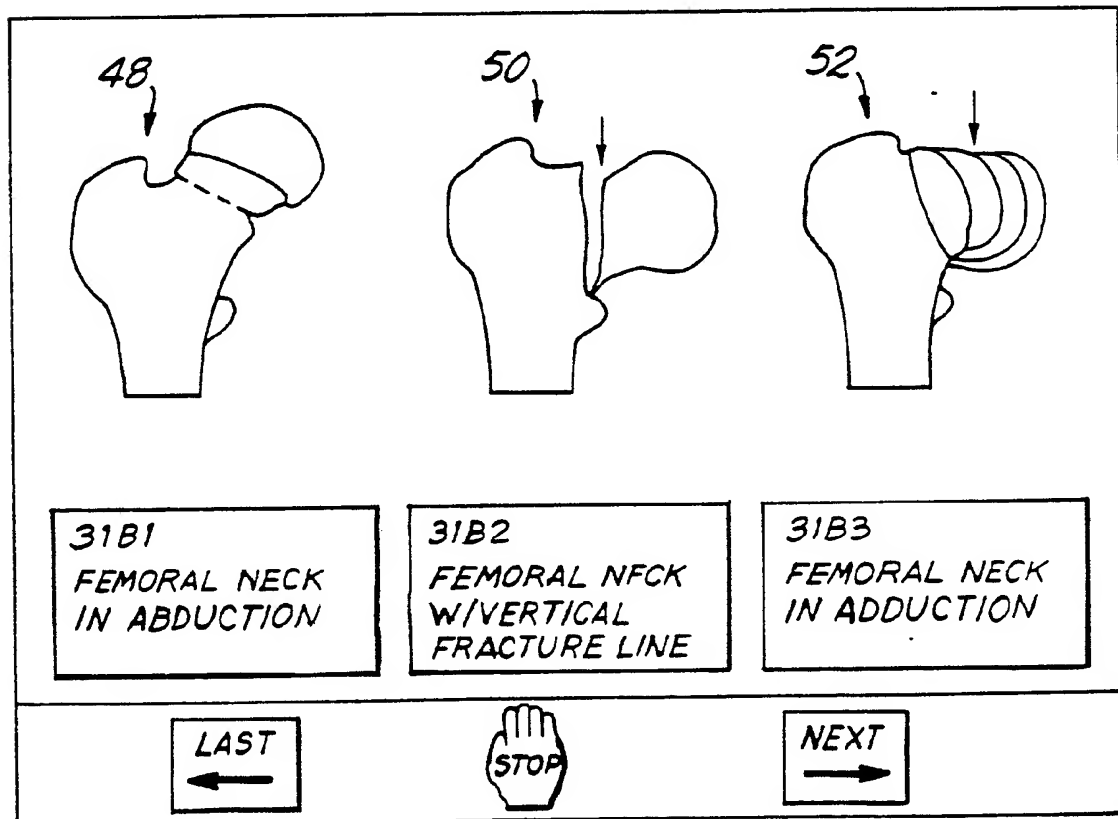


FIG. 5

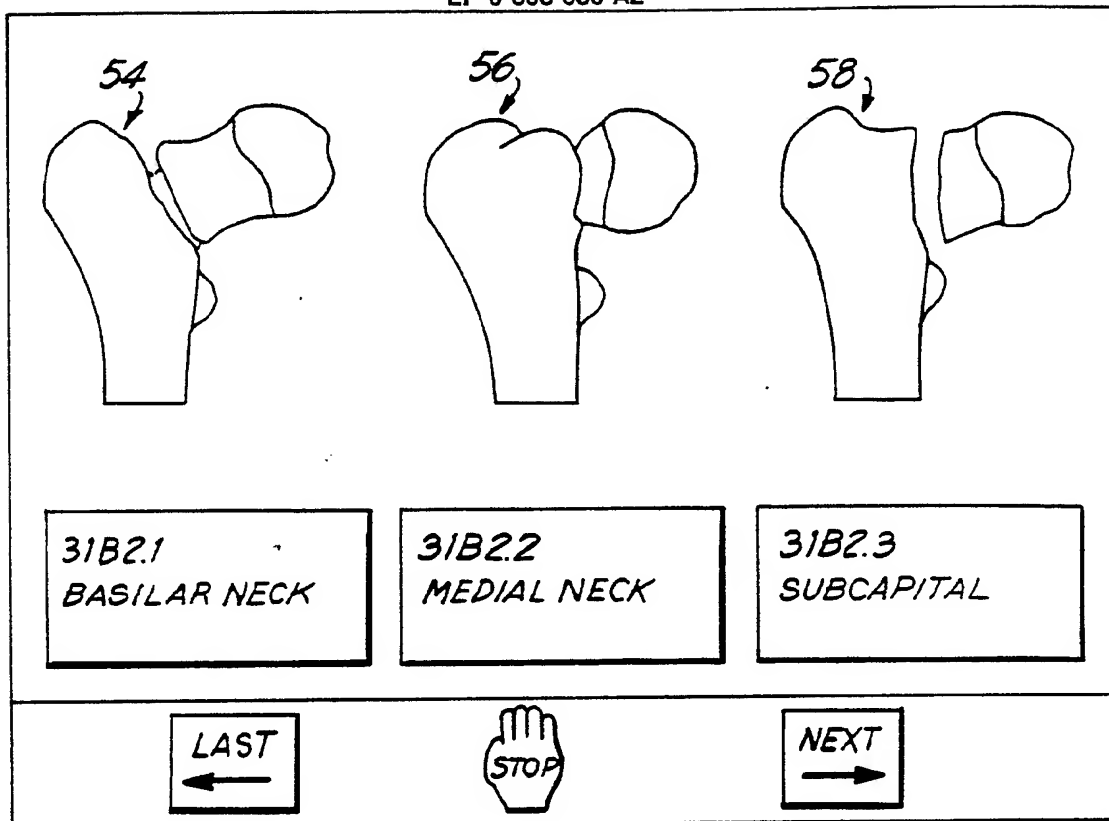


FIG. 6

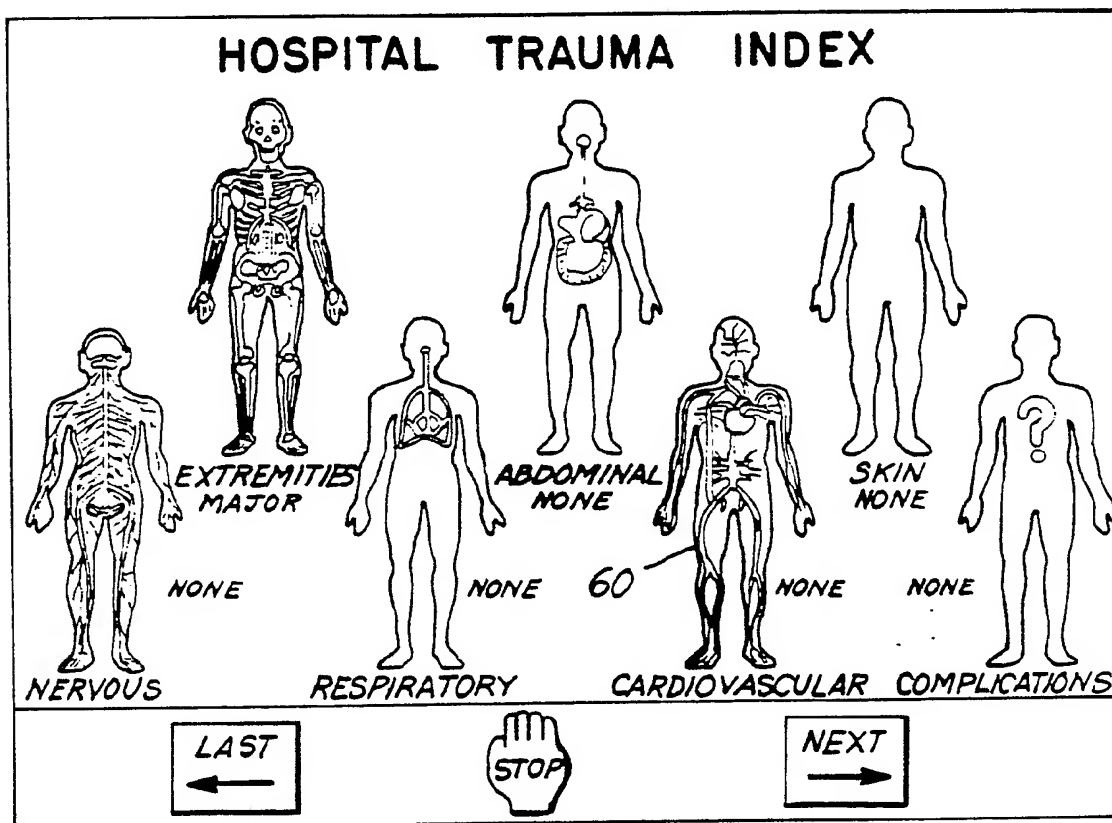


FIG. 7

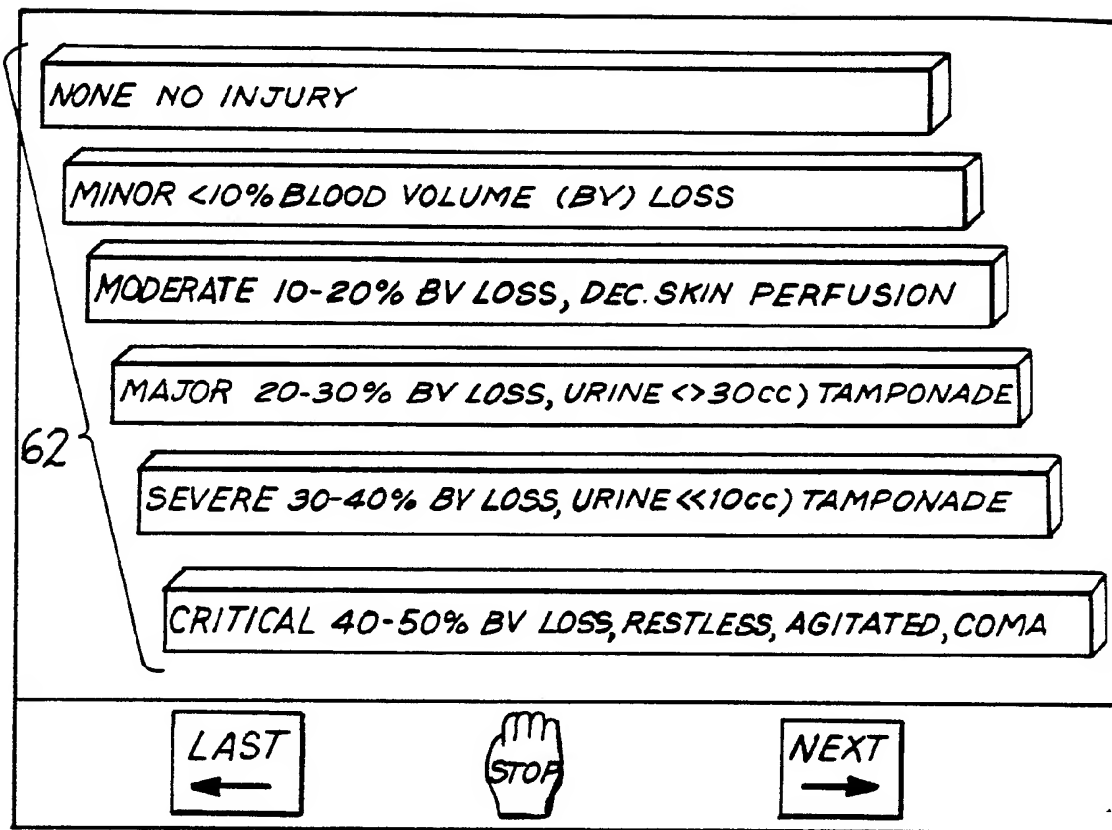


FIG. 8

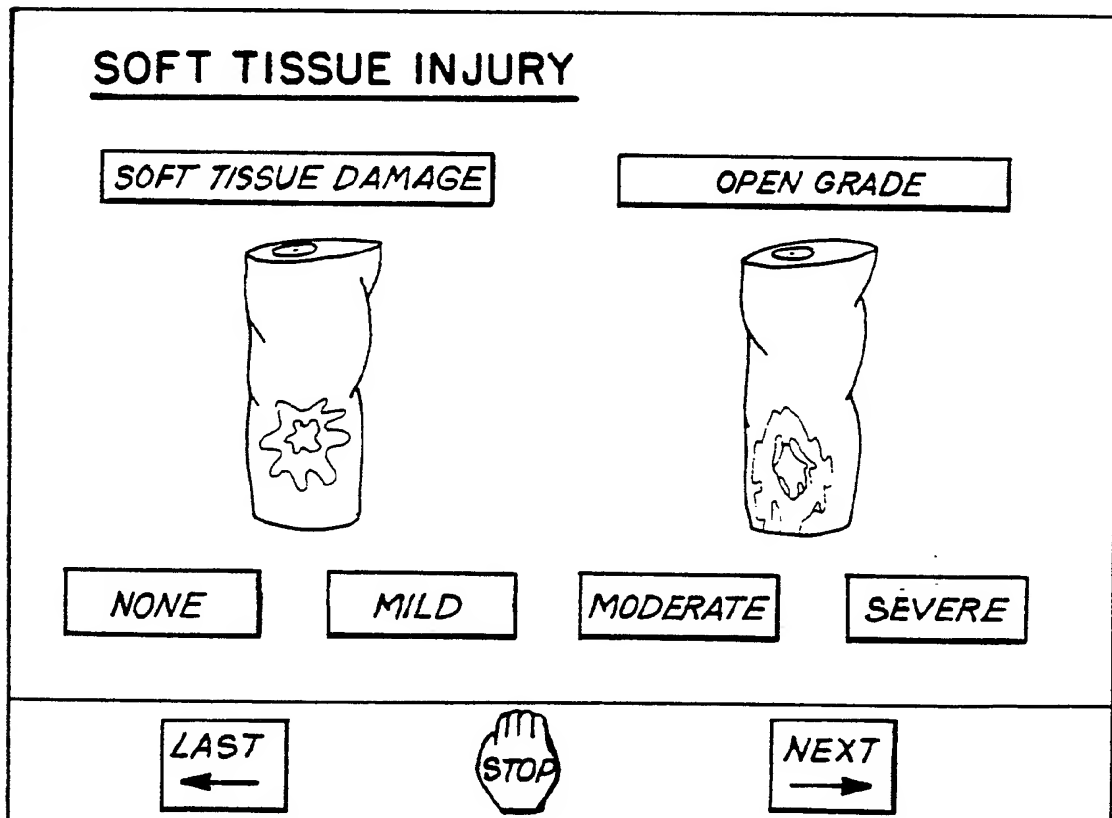


FIG. 9






 <u>60 K</u>	 <u>35</u>	 <u>170 CM</u>	 <u>MALE</u>
1	2	3	
4	5	6	
7	8	9	
*	0	#	
LAST ←		NEXT →	

FIG. 10

OCCUPATION OR LIFESTYLE
IN THE CONTEXT OF THE INJURY





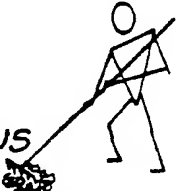
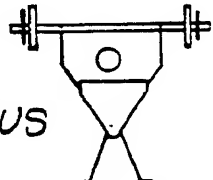

 SEDENTARY	 SEDENTARY	
 ACTIVE	 ACTIVE	
 VIGOROUS	 VIGOROUS	
LAST ←		NEXT →

FIG. 11

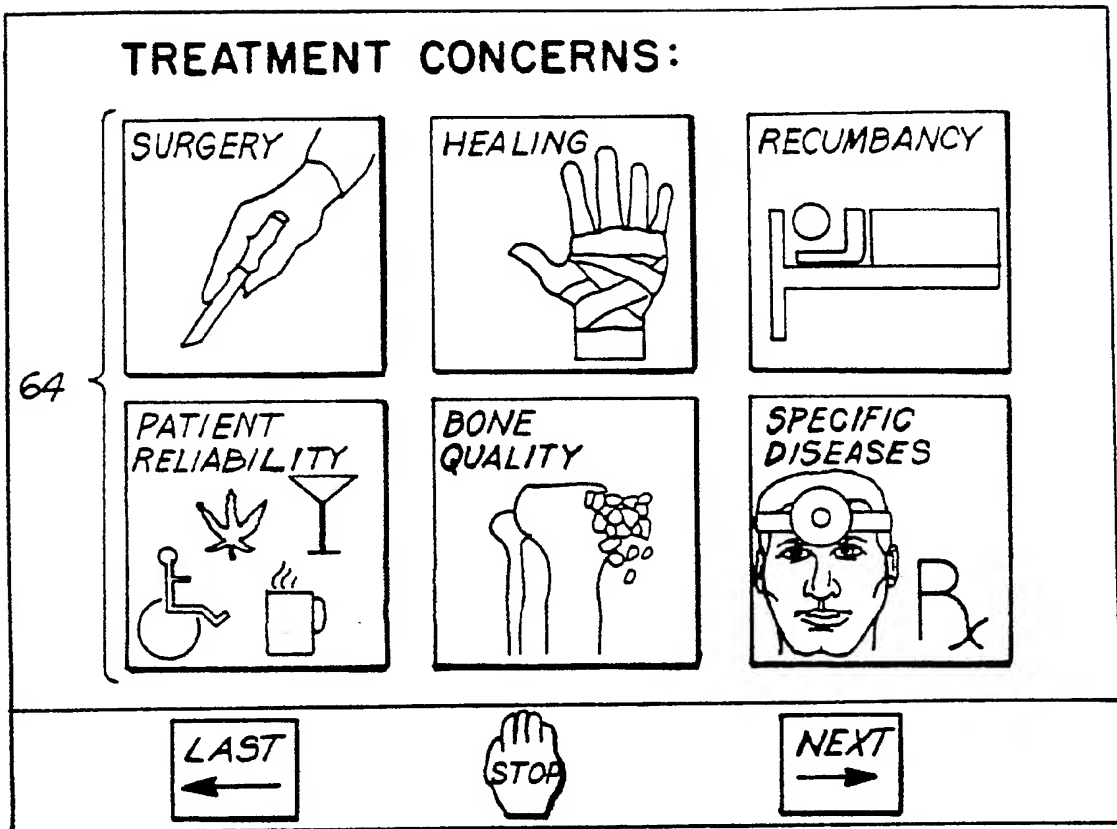


FIG. 12

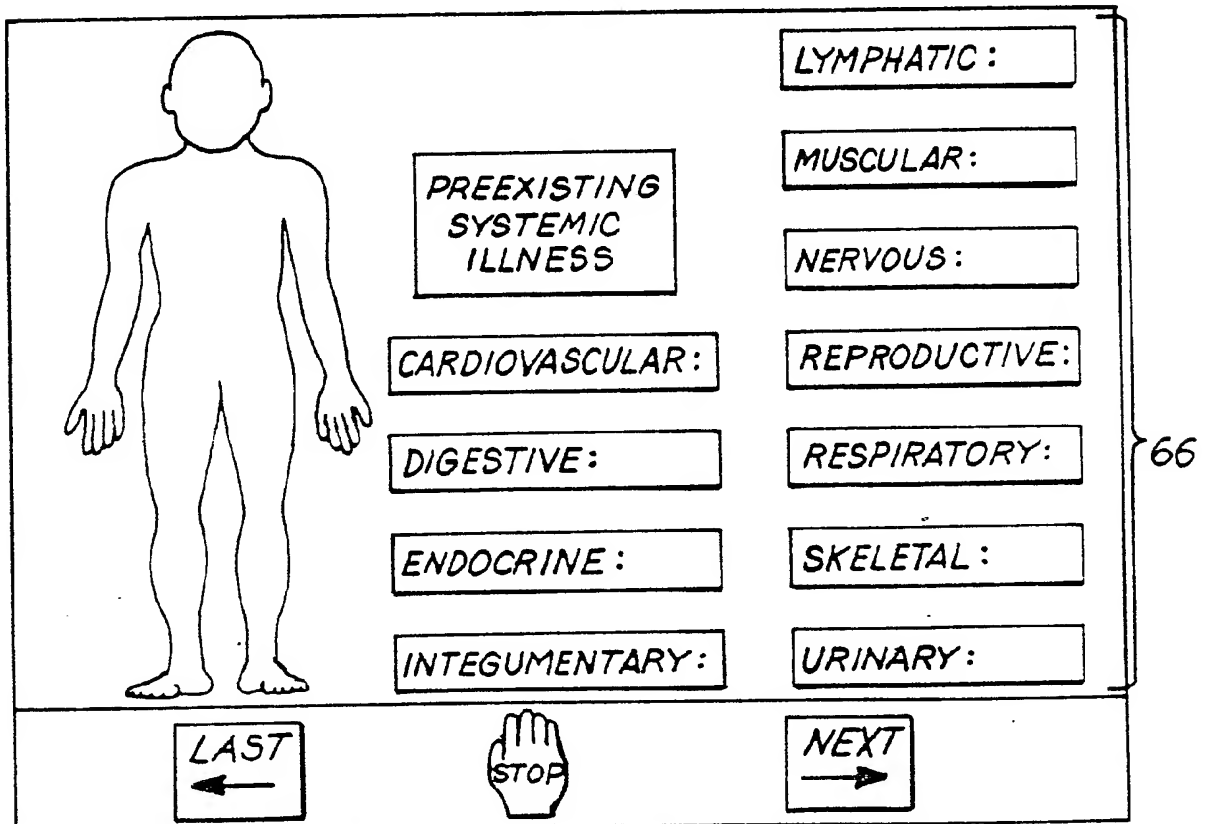


FIG. 13

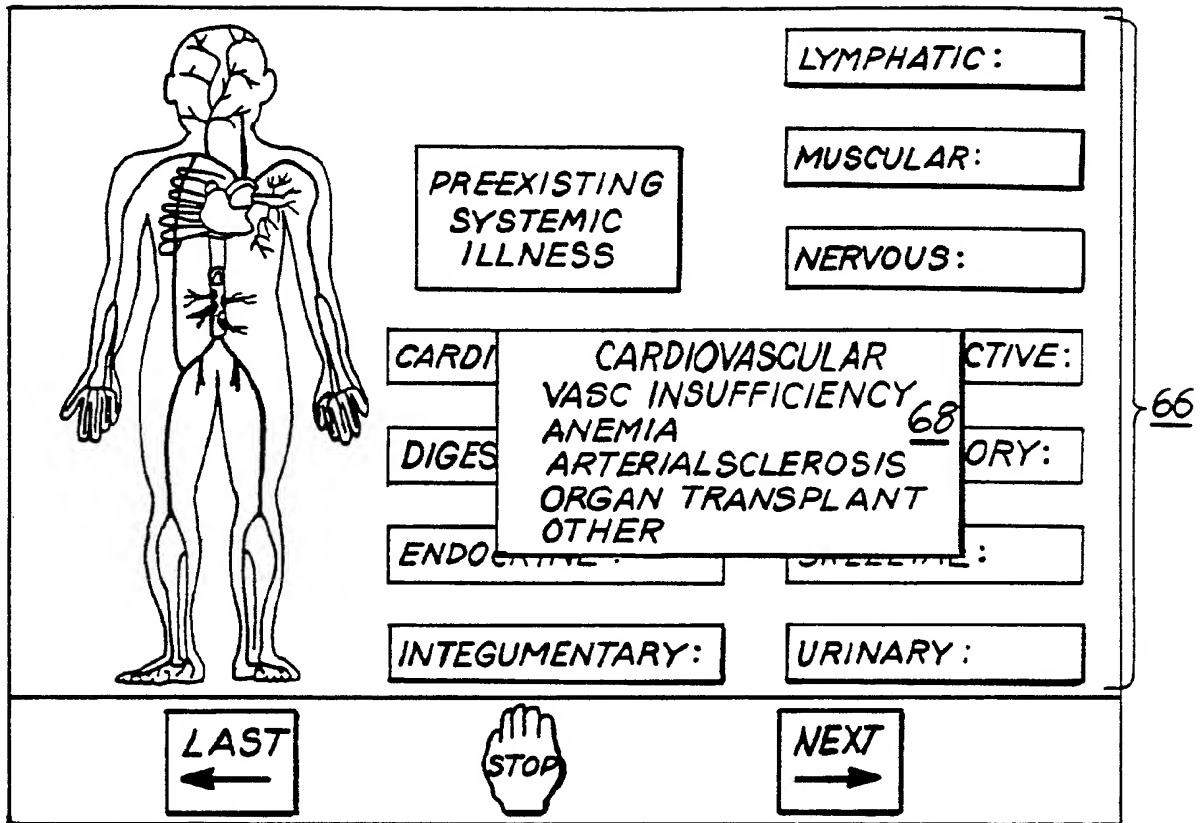


FIG. 14

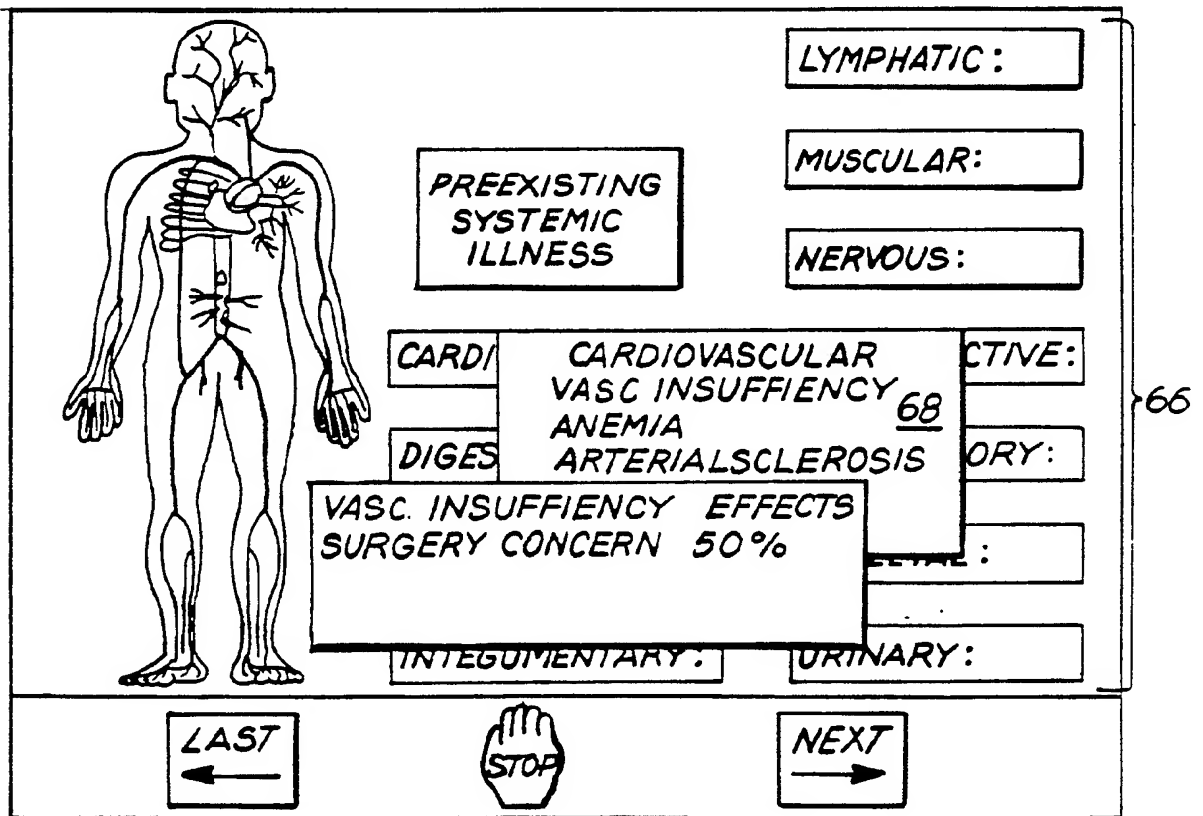


FIG. 15

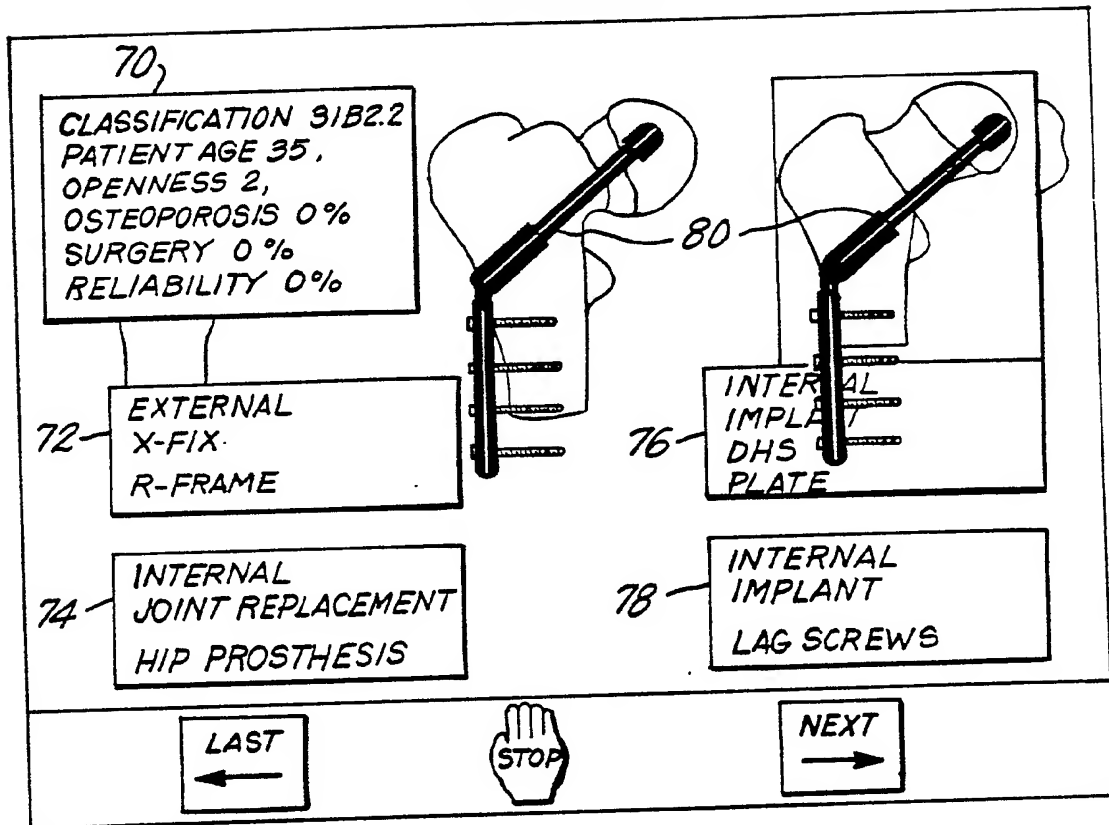


FIG. 16

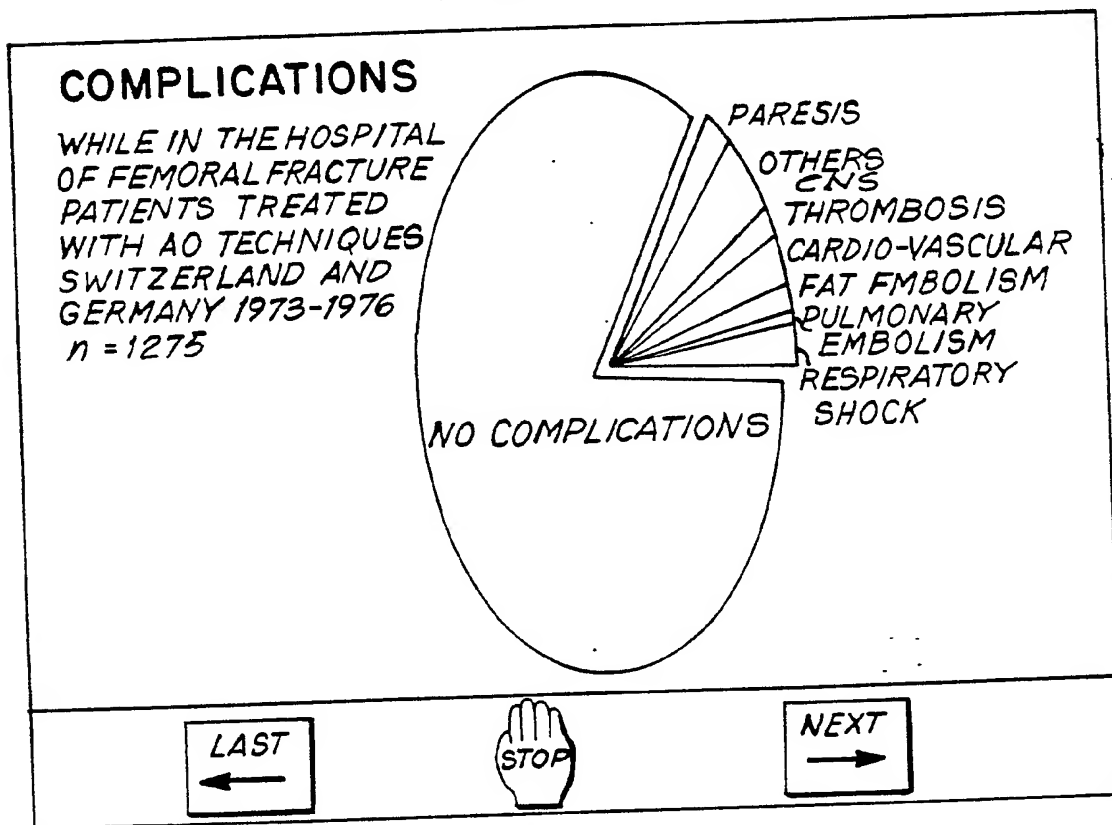


FIG. 17

<u>ADVISE FRAME</u>			
GOALS: INPUT-READ TREATMENT FINISHED		RULES: _____ _____ _____ _____ PARAMETERS: _____ _____ _____ _____ _____	
<u>PRO-CONS FRAME</u>			
SUB- GOALS: STRONGER IMMEDIATELY STABLE NON-DISRUPTING ACCURATE EASY VERSATILE FEEDBACKS COSMETIC		RULES: _____ _____ _____ _____ PARAMETERS: _____ _____ _____ _____ _____	

<u>PATIENT FRAME</u>		<u>TRAUMA FRAME</u>	
RULES: _____	PARAMETERS: _____	RULES: _____	PARAMETERS: _____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____

FIG. 18

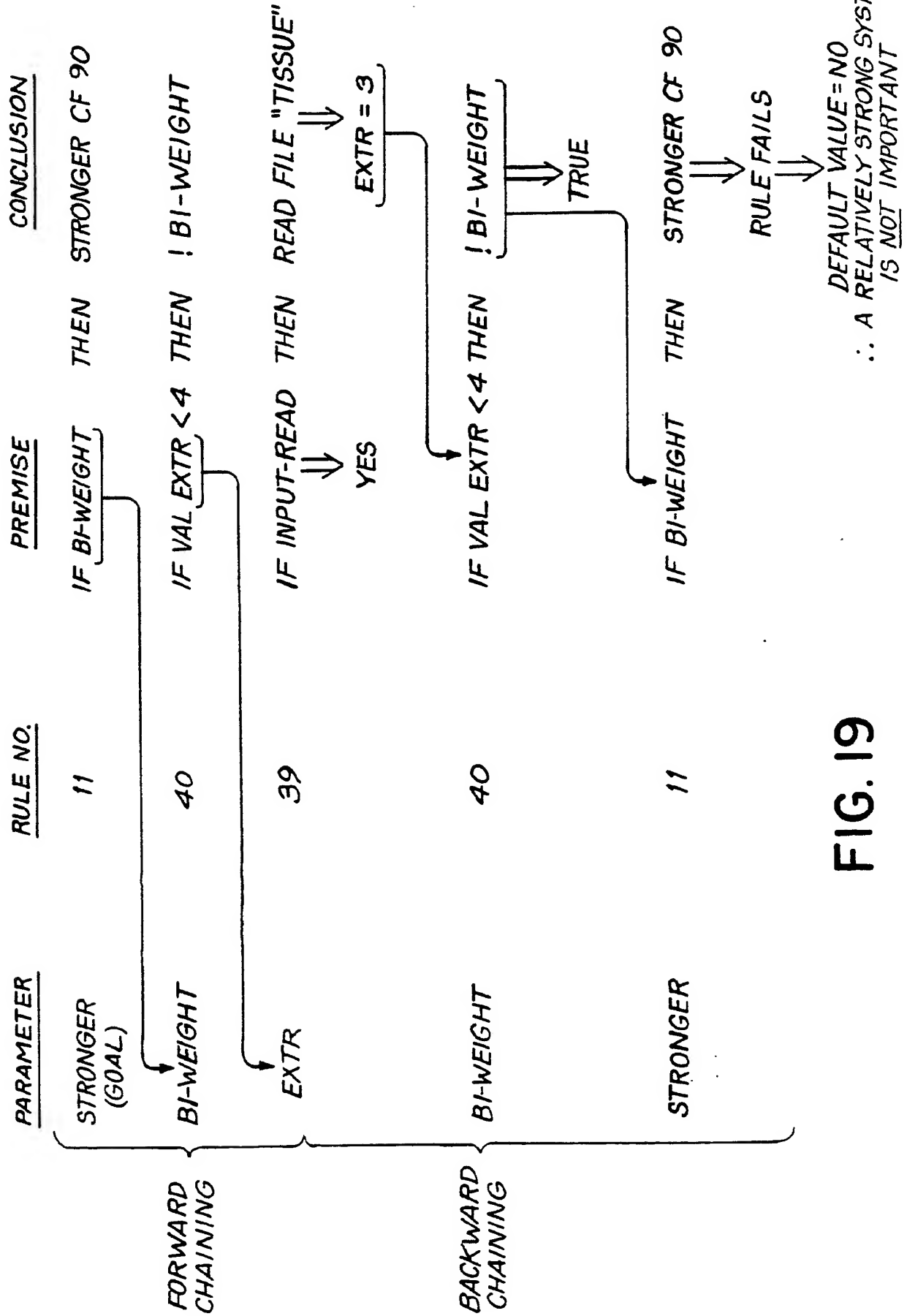


FIG. 19

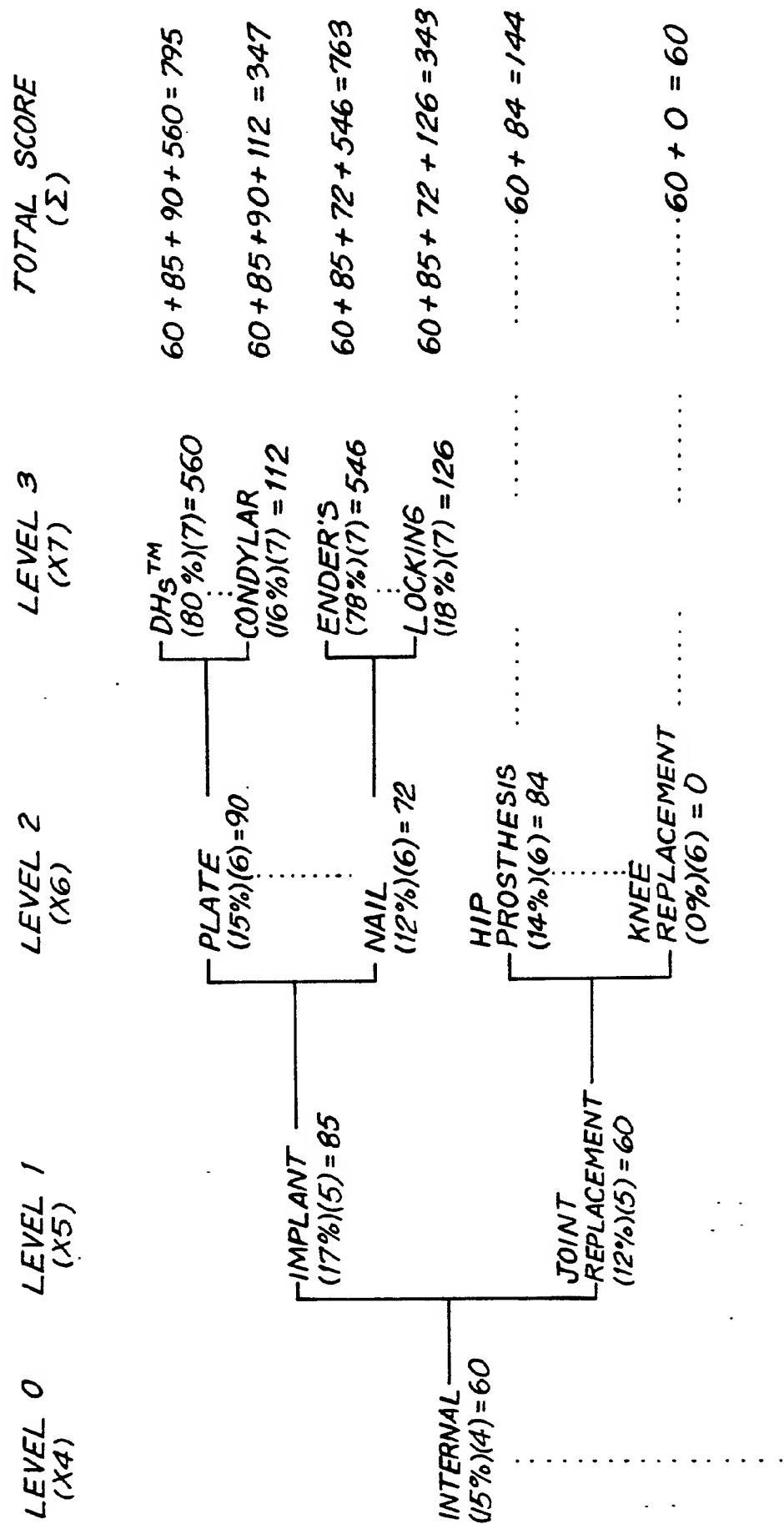


FIG. 20

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/*      COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED      */ #include
<stdio.h> #include <string.h> #include <dos.h> #include <stdlib.h> #include
<memory.h> #include <conio.h>

```

```

/***** program to determine initial information for aoca *****/

```

```

/* btrieve definitions */ #define B_OPEN 0 #define B_CLOSE 1 #define B_GETNX
#define B_GETEQ 5 #define B_GETGE 9 #define EOF_ERR 9 /* database parameters
int op; /*operation code*/ char pos_blk1[128]; /*position block*/ c
pos_blk2[128]; int buf_len; /* length of above */ char key_buf[30]; /*
buffer */ int key_num; /* key number */ int status; /* error status
char entry[30]; /* key copy */ struct

```

```

{
char class[10]; /*classification code */
char proc[30]; /* procedure suggestion */
char empty[39];
} sugg_rec; struct
{
char class[10]; /* classification code */
char artic[3]; /*yes/no articular*/
char inher[3]; /*yes,no inherently stable */
char number[2]; /* number of pieces */
char potent[3]; /*yes/no potentially stable*/
char weightb[3]; /*yes/no weightbearing */
char empty[1];
} expan_rec;

```

```

FILE *tempf; /* temporary file for data passing */

```

```

main() {
int choice,upto,stat1,stat2;

/* now prepare for touchscreen use */
info_stop(); /*clear problems info window*/
info_start(); /*initialize info window*/
setega();
initgraf(16,2,0); /* mode of 640 X 350 colors*/
fontinit();
fontld(0,"IBMROM");
set_choice();
upto = 0;
while (upto < 10) /* main while */
{
switch (upto) /* main switch */
{
case 0: /* class info */
if ( ( tempf = fopen("trauma.rd","w")) == NULL) {
fprintf(stderr,"%s couldnt open file 'trauma.rd'");
exit(1); ;
}
}
}
}

```


COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

        class_info(); /* write frac-class, #procedures*/
        break;
    case 1: /* tissue info */
        if ( ( tempf = fopen("tissue.rd","w")) == NULL) {
            fprintf(stderr,"%s couldnt open file 'tissue.rd'");
            exit(1);
        }

        part_halo_draw("c:\\clinical\\hti.pic",4);
        stat1=hti();
        part_halo_draw("c:\\clinical\\inco.pic",4);
        stat2=soft();
        break;
    case 2 :
        if ( ( tempf = fopen("patient.rd","w")) == NULL) {
            fprintf(stderr,"%s couldnt open file 'patient.rd'");
            exit(1);
        }
        part_halo_draw("c:\\clinical\\init1.pic",4);
        stat1=patient(); /* write out initial patient info */
        part_halo_draw("c:\\clinical\\lifeocc.pic",4);
        stat2=life();
        break;

    case 3 : /***** disease info */
        if ( ( tempf = fopen("disease.rd","w")) == NULL) {
            fprintf(stderr,"%s couldnt open file 'patient1.rd'");
            exit(1);
        }
        part_halo_draw("c:\\clinical\\concern.pic",4);
        stat1=concern();
        break;
    } /* end main switch */
    upto ++;
    fcloseall(); /* close all the open files */
    if (stat1 == 1 || stat2 == 1 && upto > 1) upto = upto-2; /* back a level */
    /* end main while */
    /* prepare to finish */
    initgraf(3,0,0); /* set back to 80 column with color */
    info_stop(); /* stop it */
    fcloseall(); /* close all files */
    exit();
    } /* end of main program for initial data */
/*****
class_info() {
    int choice,status;
    char data[10];

    /* get user input */
    classify(entry);
    /* open database for classification expansion */
    status = btrv(B_OPEN,pos_blk2,&expan_rec,&buf_len,"classexpan.dat",0);

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/* now get classification expansion data */
buf_len = sizeof(expan_rec);
key_num = 0; /* classification code */
strcpy(key_buf, entry); /* copy into key buffer for use */
status = btrv(B_GETEQ, pos_blk2, &expan_rec, &buf_len, key_buf, key_num);
    if (status != 0) printf("status = %i", status);
printf("\n %10s \n articular: %3s \n inherently stable: %3s",
expan_rec.class, expan_rec.artic, expan_rec.inher);
printf("\n number pieces: %2s \n potentially stable: %3s",
expan_rec.number, expan_rec.potent);
printf("\n weightbearing: %3s", expan_rec.weightb);
data_out(expan_rec.class, 10, 100, "classification");
yesno_out(expan_rec.artic, 3, 100, "articular");
yesno_out(expan_rec.inher, 3, 100, "inherently stable");
sscanf(expan_rec.number, "%i", &choice); /*convert */
numb_out(choice, 100, "number of pieces");
yesno_out(expan_rec.potent, 3, 100, "potentially stable");
yesno_out(expan_rec.weightb, 3, 100, "weight bearing");
/* open database for procedure suggestions */
status = btrv(B_OPEN, pos_blk1, &sugg_rec, &buf_len, "procsugg.dat", 0);
if (status != 0)
{
    printf("Error opening file . Status = %d", status);
    exit(0);
}
proc_sugg();

/* close database */
status = btrv(B_CLOSE, pos_blk1, &sugg_rec, &buf_len, key_buf, 0);
status = btrv(B_CLOSE, pos_blk2, &expan_rec, &buf_len, key_buf, 0);
if (status != 0)
{
    printf("Error closing file . Status = %d", status);
    exit(0);
}
printf("\n\n\n");
info_touch(1); /* turn on responses */
info_touch_res(data); /*wait to touch*/
return(0); } /* end of class_info */

/*****/
numb_out(numb, cerfac, string) /* this routine writes the number*/
/* to a file in a format PCPLUS will recognize */

int numb, cerfac; char string[];

{
    int i;

    fprintf(tempf, "\n(");
    fprintf(tempf, "%i", numb); /* print out the number*/
    fprintf(tempf, " %i)", cerfac);

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

    return; } /* end of data_out */

/*****/
data_out(string,numb,cerfac,string2) /* this routine writes out the string,
without trailing */
/* to a file in a format PCPLUS will recognize */

char string[]; int cerfac,numb; char string2[];

{
    int i;

    /* determine how many trailing blanks there are */
    while (string[numb-1] == 32 || string[numb-1] == 0)
        numb--;

    fprintf(tempf,"\n(\"");
    for (i=0; i < numb; i++)
        fprintf(tempf,"%c",string[i]); /* print out non-blank character*/
    fprintf(tempf,"\" %i)",cerfac);
    return;
} /* end of data_out */

/*****/
yesno_out(string,numb,cerfac,string2) /* this routine writes out the yesno
parameter, without trailing */
/* to a file in a format PCPLUS will recognize */

char string[]; int cerfac,numb; char string2[];

{
    int i;

    /* determine how many trailing blanks there are */
    while (string[numb-1] == 32 )
        numb--;

    fprintf(tempf,"\n(");
    for (i=0; i < numb; i++)
        fprintf(tempf,"%c",string[i]); /* print out non-blank character*/
    fprintf(tempf," %i)",cerfac);
    return;
} /* end of yesno_out */

/*****/
int halo_draw(name2) /* begin definition of halo draw,direct to ega mem */

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

name2[];

{ unsigned add,mode ; /*ega ports*/ char color; int plane,status; /* first se
write mode to 0*/ add = 0x3CE; mode = 0x3CF; outp(add,5); outp(mode,0);

/* now set data mask to full*/ outp(add,0x8); outp(mode,0xFF); /* now loop fo
each plane */ color = 1; for ( plane=0; plane<4; plane++)
{
    /* set the map mask for this plane */
    add = 0x3C4; /* address for map mask */
    mode = 0x3C5;
    outp(add,2);
    outp(mode,color);

    status=draw_plane(name2,plane); /* draw the plane */
    if (status!=0) break; /*dont continue if picture is not there */
    color = color << 1; /* next color */
}
return(status); } /* end of halo_draw */
/*****

int draw_plane(name2,plane) /* routine to draw a plane of data */ char name
int plane; {

char far *pointer; char far *plength; static FILE *stream; unsigned char
line[512]; /* to hold line of data */ int datalen,numread; register short i
loop,count;

count = 0; FP_SEG(plength) = 0xA000; FP_OFF(plength) = 80 * 350; /* end of pl
*/

if (plane == 0 ) {
    /* open picture file */
    if ((stream = fopen(name2,"rb")) == NULL) {
        printf("\r file %.30s not found",name2);
        return(1);
    }
    numread = fread(line,1,512,stream); /* read the first block */
    if ((line[0] != 0x41) || (line[1] != 0x48) )/* sure dr halo */
        {initgraf(3,0,0); /* back to text mode */
        fprintf(stderr,"error - not a dr halo file ");
        initgraf(3,0,0); /* set back to 80 column with color */
        info_stop();/*stop it */
        exit(1);}
    /* mode of 640 X 350 16 colors */
    count = 12; /* need to skip past header file */
    } if (plane != 0 ) numread = fread(line,1,512,stream); /* just read in fi
line*/

/* set pointer to beginning of plane */

/* screen begins at a000, 0000 */ FP_SEG(pointer) = 0xA000; FP_OFF(pointer)
0x0000;

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

while(numread == 512 )    /* for each block */
{
    while((count < 512) || (line[count] != 128)) /* for each byte */
    {
        if (line[count]==0) break; /*end of block */
        if (line[count]==0x80) break; /*end of block */
        datalen = line[count]&0x7F;
        if(line[count]&0x80)    /* duplication symbol */
        {
            count++;
            for (loop=datalen; loop > 0 ; loop--)
            {
                *pointer = line[count];
                pointer++;
            } /* end looping */
            count++; /* done with this data */
        } /* end duplication loop */
        else /* no duplication */
        {
            count++;
            for (loop=datalen;loop > 0;loop--)
            {
                *pointer = line[count];
                pointer++;
                count++; /*done with this data */
            } /* end looping */
        } /* end no duplications */
    } /* end block */
    numread = 0;
    if (pointer < plength) { /* test for full screen */
        numread = fread(line,1,512,stream); /* read the next line */
        count = 0;
    }
} /* end of the plane */ if (plane == 3) fclose(stream); return(0); } /*
of plane_draw */
/*****
proc_sugg()
/* now get procedure suggestion information */
/* check for blanket suggestions at all levels */ {
    int count;
    char savbuf[30];

    buf_len = sizeof(sugg_rec);
    for (count=9; count>0 ; count--) /* check for blanket suggestions */
    {
        key_num = 1; /* by classification - all 10 characters*/
        if (count==9) /*first time */
        {
            status = btrv(B_GETGE,pos_blk1,&sugg_rec,&buf_len,entry,key_num,
            if (status != 0) printf("\n %i",status);
            strcpy(key_buf,entry); /* copy into key buffer for later compare
            strcpy(savbuf,entry);
        } /*end of first time search */
        else {

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

        status = btrv(B_GETEQ,pos_blk1,&sugg_rec,&buf_len,key_buf,key_num
    )
    while(status ==0 && (strcmp(key_buf,entry)== 0)) /* while key not cha
*/
    {
        /*printf ("\n %.10s %.30s",sugg_rec.class,sugg_rec.proc);*/
        data_out(sugg_rec.proc,30,75,"suggestion");
        status = btrv(B_GETNX,pos_blk1,&sugg_rec,&buf_len,key_buf,key_num
    ) /* end of key not changed while */
    while (savbuf[count]!='X') count--; /*already checked */
    savbuf[count] = 'X';
    strcpy(key_buf,savbuf);
    strcpy(entry,savbuf);
    } /* end of count for */ } /* end of proc_sugg */

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/*      COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED      */ #include
<dos.h> /* this contains subroutines for the clinical gathering */
static struct pats {
    int age;
    char sex[8];
    int weight ;
    int height ;
    char wht_unt[5];
    char ht_unt[5];
};

int patient() /* gather patient stats*/
{
    static char datal[80]={1,1,21,4,36,2,1,55,4,74,3,1,6,4,21,
4,1,36,4,55,5,16,31,20,45,6,6,16,10,31,7,6,31,10,45,8,6,45,10,62,
9,10,16,13,31,10,10,31,13,45,11,10,45,13,62,12,13,16,16,31,
13,13,31,16,45,14,13,45,16,62,
15,16,16,20,31,16,16,45,20,62};
    char data[115];
    int rs,rs1,col1,col2,row1,row2;
    int rs2,rs3,rs4;
    static struct pats pat_info = {
        35,"MALE",60,170,"K  ","cm "};

    load_choice();
    draw_all(pat_info);
    strncpy(data,datal,80);
    info_load_key(80,data); /* set up key table*/
    /*draw initial values */
    info_touch(3); /*enable key table response */
    while((rs=info_touch_res(data)) < 21) /* get response*/
    {
        info_touch(0); /*touch off*/
        /* based on this response do some action */
        if (rs > 0 && rs < 5)
        {
            switch(rs)
            {
                case 1: /*age*/
                    draw_unit("Yr","");
                    info_touch(3); /*touch on */
                    rs1=info_touch_res(data); /*get response*/
                    if (rs1 >4 && rs1 < 15)
                    {
                        pat_info.age = (rs1-5)*100;
                        rs2=info_touch_res(data); /*get response*/
                        if (rs2 >4 && rs2 < 15)
                        {
                            pat_info.age += (rs2-5)*10;
                        }
                    }
                }
            }
        }
    }

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

        rs3=info_touch_res(data); /*get response*/
        if (rs3 >4 && rs3 < 15)
        {
            pat_info.age += rs3-5;
        }
        else pat_info.age=pat_info.age/10;
    }
    draw_all(pat_info);
    break;

case 2: /*sex*/
    draw_unit("M","F");
    info_touch(3); /*touch on */
    rs1=info_touch_res(data); /* get response*/
    if (rs1 == 15) strcpy(pat_info.sex,"MALE");
    if (rs1 == 16) strcpy(pat_info.sex,"FEMALE");
    draw_all(pat_info);

    break;

case 3: /*weight*/
    draw_unit("Lb","K");
    info_touch(3); /*touch on */
    rs1=info_touch_res(data); /*get response*/
    if (rs1 >4 && rs1 < 15)
    {
        pat_info.weight = (rs1-5)*100;
        rs2=info_touch_res(data); /*get response*/
        if (rs2 >4 && rs2 < 15)
        {
            pat_info.weight += (rs2-5)*10;
            rs3=info_touch_res(data); /*get response*/
            if (rs3 >4 && rs3 < 15)
            {
                pat_info.weight += rs3-5;
                rs4=info_touch_res(data); /*get response*/
                if (rs4 == 15) strcpy(pat_info.wht_unt,"lbs");
                if (rs4 == 16) strcpy(pat_info.wht_unt,"K ");
            }
            else pat_info.weight=pat_info.weight/10;
            if (rs3 == 15) strcpy(pat_info.wht_unt,"lbs");
            if (rs3 == 16) strcpy(pat_info.wht_unt,"K ");
        }
        if (rs2 == 15) strcpy(pat_info.wht_unt,"lbs");
        if (rs2 == 16) strcpy(pat_info.wht_unt,"K ");
    }
    draw_all(pat_info);
    break;

case 4: /*height*/
    draw_unit("In","Cm");
    info_touch(3); /*touch on */
    rs1=info_touch_res(data); /*get response*/

```


COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

    if (rs1 >4 && rs1 < 15)
    {
        pat_info.height = (rs1-5)*100;
        rs2=info_touch_res(data); /*get response*/
        if (rs2 >4 && rs2 < 15)
        {
            pat_info.height += (rs2-5)*10;
            rs3=info_touch_res(data); /*get response*/
            if (rs3 >4 && rs3 < 15)
            {
                pat_info.height += rs3-5;
                rs4=info_touch_res(data); /*get response*/
                if (rs4 == 16) strcpy(pat_info.ht_unt,"cm ");
                if (rs4 == 15) strcpy(pat_info.ht_unt,"in ");
            }
            else pat_info.height=pat_info.height/10;
            if (rs3 == 16) strcpy(pat_info.ht_unt,"cm ");
            if (rs3 == 15) strcpy(pat_info.ht_unt,"in ");
        }
        if (rs2 == 16) strcpy(pat_info.ht_unt,"cm ");
        if (rs2 == 15) strcpy(pat_info.ht_unt,"in ");
    }
    draw_all(pat_info);
    break;
} /* end switch */
} /* end if */
info_touch(3); /*touch on */
} /* end while */
/* do conversion if necessary */
if (strcmp(pat_info.wht_unt,"lbs") == 0) pat_info.weight /= 2.2;
if (strcmp(pat_info.ht_unt,"in ") == 0) pat_info.height *= 2.54;
numb_out(pat_info.age,100,"age");
data_out(pat_info.sex,6,100,"sex");
numb_out(pat_info.weight,100,"weight");
numb_out(pat_info.height,100,"height");
if (rs == 40) return(1);
if (rs == 41) return(0);
if (rs == 42) mayexit();
} /* end of patient */
/*****
draw_unit(st1,st2) /* this function draws the unit choices on the screen */

char st1[],st2[]; {
    /* draw the values at this time */
    grrtulc(172,258,20,10,10); /*clear some space */
    fcurloc(175,264); /*move cursor*/
    fcursprt(0,st1,0,3,0);
    grrtulc(420,258,20,10,10); /*clear some space */
    fcurloc(425,264); /*move cursor*/
    fcursprt(0,st2,0,3,0);
    return; }

/***** in
life() /* gather life style info */

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

{
static char datal[15]={1,4,1,8,80,2,8,1,16,80,3,16,1,23,80};
char data[70];
static char lifeocc[40]=' ','\0';
int rs,rs1,col1,col2,row1,row2;

load_choice();
strncpy(data,datal,15);
info_load_key(15,data);/*set up key table*/
/* draw initial values */
info_touch(3); /*enable key table response */
while((rs=info_touch_res(data)) < 40 ) /* get response*/
{
    if (rs > 0 && rs < 4)
    {
        /* based on this response do some action */
        typedot(1); /* xor mode*/
        col1 = 300;
        col2 = 350;
        row1 = 14*(datal[rs*5-4])+32;
        row2 = row1 + 8;
        grbxfill(col1,row1,col2,row2,4);
        typedot(0);
        if (rs == 1) strcpy(lifeocc,"SEDENTARY");
        if (rs == 2) strcpy(lifeocc,"ACTIVE");
        if (rs == 3) strcpy(lifeocc,"VIGOROUS");
        } /* end if */
    } /* end while*/
data_out(lifeocc,9,100,"life-occ style");
if (rs == 40) return(1);
if (rs == 41) return(0);
if (rs == 42) mayexit();
return; } /* end of life */

/*****

int soft() /* gather soft tissue info */

{
static char datal[30]={1,5,13,9,33,2,5,45,9,67,
    5,17,7,23,25,6,17,25,23,39,7,17,39,23,57,8,17,57,23,70};
char data[70];
static int open={0};
static int damage={0};
int rs,rs1,col1,col2,row1,row2,temp1,temp2;

load_choice();
/* first set to proper levels */
temp2 = open;
open = -1;
temp1 = damage;
damage = -1; /* this is how it is originally drawn */
set_soft(&damage,temp1,&open,temp2);

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

strncpy(data,datal,30);
info_load_key(30,data);/*set up key table*/
/* draw initial values */
info_touch(3); /*enable key table response */
while((rs=info_touch_res(data)) < 40) /* get response*/
{
    if (rs > 0 && rs < 9)
    {
        switch (rs)
        {
            case 1 :
            case 2 :
                /* based on this response do some action */
                typedot(1); /* xor mode*/
                col1 = 8*datal[rs*5-3]-1;
                col2 = 8*datal[rs*5-1]-1;
                row1 = 14*datal[rs*5-4]-1;
                row2 = row1 + 10;
                grbxfill(col1,row1,col2,row2,4);
                typedot(0);
                /* damage or openness chosen */
                rs1 = info_touch_res(data);

                if (rs1 > 4 && rs1 < 9)
                {
                    if (rs==1) set_soft(&damage,rs1-5,&open,open);
                    if (rs==2) set_soft(&damage,damage,&open,rs1 - 5);
                } /* end if */

                typedot(1); /* xor mode*/
                col1 = 8*datal[rs*5-3]-1;
                col2 = 8*datal[rs*5-1]-1;
                row1 = 14*datal[rs*5-4]-1;
                row2 = row1 + 10;
                grbxfill(col1,row1,col2,row2,4);
                typedot(0);

                break;
            } /* end switch */
        } /* end if */
    } /* end while*/
    numb_out(damage,100,"damage");
    numb_out(open,100,"open");
    if (rs == 40) return(1);
    if (rs == 41) return(0);
    if (rs == 42) mayexit();
    return; } /* end of soft */
/*****
draw_all(pat_info)
    struct pats pat_info;

{
    char buff[10];

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/* draw the values at this time */

grrtllc(60,78,50,12,10);/*clear some space */
fcurloc(62,76); /*move cursor*/
sprintf(buff,"%i",pat_info.weight);
fcursprt(0,buff,0,3,0);
grrtllc(85,78,30,12,10);/*clear some space */
fcurloc(87,76); /*move cursor*/
fcursprt(0,pat_info.wht_unt,0,3,0);

grrtllc(200,78,50,12,10);/*clear some space */
fcurloc(202,76); /*move cursor*/
sprintf(buff,"%i",pat_info.age);
fcursprt(0,buff,0,3,0);

grrtllc(340,78,50,12,10);/*clear some space */
fcurloc(342,76); /*move cursor*/
sprintf(buff,"%i",pat_info.height);
fcursprt(0,buff,0,3,0);

grrtllc(365,78,50,12,10);/*clear some space */
fcurloc(367,76); /*move cursor*/
fcursprt(0,pat_info.ht_unt,0,3,0);

grrtllc(480,78,50,12,10);/*clear some space */
fcurloc(482,76); /*move cursor*/
fcursprt(0,pat_info.sex,0,3,0);

return;

} /* end of draw_all */
/******/ int
set_soft(damage,newd,open,newo)/* to change soft values and change screen */
*damage,newd,*open,newo;
{
char name[40]; int i;

strcpy(name,"\\clinical\\incod");
if (newd == 0 ) strcat(name,"0");
if (newd == 1 ) strcat(name,"1");
if (newd == 2 ) strcat(name,"2");
if (newd == 3 ) strcat(name,"3");
strcat(name,".cut");
if (*damage!=newd ) cut_halo_draw(name,112,118);
strcpy(name,"c:\\clinical\\incod");
if (newo == 0 ) strcat(name,"0");

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```
if (newo == 1 ) strcat(name,"1");
if (newo == 2 ) strcat(name,"2");
if (newo == 3 ) strcat(name,"3");
strcat(name,".cut");
if (*open!=newo ) cut_halo_draw(name,416,114);
*damage =newd;
*open = newo;
return(0);
)/* end of set_soft */
```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/*****
COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED          */ /* Routines to a
the user to pick the classification */

```

```

#include <string.h>

```

```

/* btrieve definitions */ #define TRUE 1 #define FALSE 0 #define B_OPEN 0 #de
B_CLOSE 1 #define B_GETNX 6 #define B_GETEQ 5 #define B_GETGT 8 #define B_GET
#define EOF_ERR 9 /* database parameters */ extern int op; /*operati
code*/ extern char pos_blk1[128]; /*position block*/ extern char pos_blk2[12
extern int buf_len; /* length of above */ extern char key_buf[30]; /* k
buffer */ extern int key_num; /* key number */ extern int status; /*
error status */ extern char entry[]; /* key copy */ struct { /* for annotat
*/

```

```

    char class[10];
    char string[128];
    char garb;
} annotate;

```

```

classify(entry) /*return the classification code chosen */ char entry[];

```

```

{
    static char data1[15]={1,1,1,20,25,2,1,26,20,57,3,1,58,20,80};
    char data[100];
    static char data3[40]={1,5,37,8,43,11,5,50,8,57,2,9,37,11,43,12,8,51,11,5
3,11,40,16,47,13,11,47,16,54,4,16,40,20,46,14,16,46,20,53};
    static char data4[15]={1,1,1,6,80,2,7,1,15,80,3,16,1,22,80};
    int rs,rs1,col1,col2,row1,row2;
    int rs2,rs3,rs4;
    int notdone;
    int level;
    int s,len; /*status */
    char name[40],namel[40],temp[10]; /* for file name passing */
    /* initialize some settings */
    level = 1;
    notdone = TRUE;
    /*draw initial values */
    entry[0]='\0';
    strcpy(namel,"CL");
    /* open database for classification annotation */
    status = btrv(B_OPEN,pos_blk2,&annotate,&buf_len,"fracture.dat",0);
    if (status != 0) printf("error opening fracture file ");
    while (notdone)
    {
        if (level < 6 ) /*display the picture if possible */
        {
            strcpy(name,"c:\\clinical\\");
            strcat(name,namel);
            strcat(name,".pic");
            /* display picture name*/
            s=part_halo_draw(name,4);

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

        annot_it(entry,level); /* draw any annotation available */
    }
    switch (level)
    {
        case 1: /*bone level*/
            load_choice();
            strncpy(data,data3,40);
            info_load_key(40,data); /* set up key table*/
            info_touch(3); /*enable key table response */
            while((rs=info_touch_res(data))==0); /* short loop get re-
sponse*/
            info_touch(0);
            if (rs==11) strcpy(temp,"1");
            if (rs==1) strcpy(temp,"1");
            if (rs==12) strcpy(temp,"2");
            if (rs==2) strcpy(temp,"2");
            if (rs==13) strcpy(temp,"3");
            if (rs==3) strcpy(temp,"3");
            if (rs==14) strcpy(temp,"4");
            if (rs==4) strcpy(temp,"4");
            if (rs!=40 && rs !=0) level++; /*if not return,increase level
            if (rs==41)
            {
                strcpy(temp,"XXXXXXXXXX");
                notdone=FALSE;
            }
            if (rs==42) mayexit();
            break;
        case 2: /* bone section level*/
            load_choice();
            strncpy(data,data4,15);
            info_load_key(15,data); /* set up key table*/
            info_touch(3); /*enable key table response */
            while((rs=info_touch_res(data))==0); /* short loop get re-
sponse*/
            info_touch(0);
            if (rs==1) strcpy(temp,"1");
            if (rs==2) strcpy(temp,"2");
            if (rs==3) strcpy(temp,"3");
            if (rs!=40 && rs !=0) level++; /*if not return,increase level
            if (rs==40)
            {
                level--; /*if return,decrease level*/
                namel[strlen(namel)-1]='\0'; /*null instead of last */
                entry[strlen(entry)-1]='\0'; /*null instead of last */
            }
            if (rs==41)
            {
                strcpy(temp,"XXXXXXXXXX");
                notdone=FALSE;
            }
            if (rs==42) mayexit();
            break;
    }

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

case 3:      /*ABC level*/
load_choice();
strncpy(data,data1,15);
info_load_key(15,data); /* set up key table*/
info_touch(3); /*enable key table response */
while((rs=info_touch_res(data))==0); /* short loop get re-
sponse*/
info_touch(0);
if (rs==1) strcpy(temp,"A");
if (rs==2) strcpy(temp,"B");
if (rs==3) strcpy(temp,"C");
if (rs==40)
{
    level--; /*if return,decrease level*/
    namel[strlen(namel)-1]='\0'; /*null instead of last */
    entry[strlen(entry)-1]='\0'; /*null instead of last */
}
if (rs!=40 && rs !=0) level++; /*if not return,increase level
if (rs==41)
{
    strcpy(temp,"XXXXXXXX");
    notdone=FALSE;
}
if (rs==42) mayexit();
break;

case 4:      /*before point*/
load_choice();
strncpy(data,data1,15);
info_load_key(15,data); /* set up key table*/
info_touch(3); /*enable key table response */
while((rs=info_touch_res(data))==0); /* short loop get re-
sponse*/
info_touch(0);
if (rs==1) strcpy(temp,"1");
if (rs==2) strcpy(temp,"2");
if (rs==3) strcpy(temp,"3");
if (rs==40)
{
    level--; /*if return,decrease level*/
    namel[strlen(namel)-1]='\0'; /*null instead of last */
    entry[strlen(entry)-1]='\0'; /*null instead of last */
}
if (rs!=40 && rs !=0) level++; /*if not return,increase level
if (rs==41)
{
    strcpy(temp,"XXXXXXXX");
    notdone=FALSE;
}
if (rs==42) mayexit();
break;

case 5:      /*after point*/
load_choice();
strncpy(data,data1,15);

```


COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

info_load_key(15,data); /* set up key table*/
info_touch(3); /*enable key table response */
while((rs=info_touch_res(data))==0); /* short loop get re-
sponse*/
info_touch(0);
if (rs > 0 && rs < 4) strcat(entry,".");
if (rs==1) strcpy(temp,"1");
if (rs==2) strcpy(temp,"2");
if (rs==3) strcpy(temp,"3");
if (rs==40)
{
    level--; /*if return,decrease level*/
    namel[strlen(namel)-1]='\0'; /*null instead of last */
    entry[strlen(entry)-1]='\0'; /*null instead of last */
}
if (rs!=40 && rs !=0) level++;
if (rs==41)
{
    strcpy(temp,"XXXXX");
    notdone=FALSE;
}
if (rs==42) mayexit();
break;
case 6: /*last chance to change */
    remove_box(namel[6]);
    load_choice();
    info_touch(3); /*enable key table response */
    while((rs=info_touch_res(data))==0); /* short loop get re-
sponse*/
    info_touch(0);
    if (rs==40)
    {
        level--; /*if return,decrease level*/
        namel[strlen(namel)-1]='\0'; /*null instead of last */
        entry[strlen(entry)-2]='\0'; /*null instead of last */
    }
    if (rs==41)
    {
        strcpy(temp,"XXXXX");
        notdone=FALSE;
        level++;
    }
    if (rs==42) mayexit();
    if (entry[0] != '3' ) force_now(&level,entry,namel); /* force
femur choice */
    break;
} /*end switch */
if (rs < 40 && rs > 0 && level < 7) /*update them*/
{
    strcat(entry,temp);
    strcat(namel,temp);
}
} /* end notdone while */
/* now pad the result with blanks */

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

len = strlen(entry);
for (level=len;level < 10;level++)
    entry[level]= ' ';

/* close database */
status = btrv(B_CLOSE,pos_blk1,&annotate,&buf_len,key_buf,0);

return(0);
} /* end of classify */

/*****

force_now(level,entry,namel) /* this routine forces a femur choice */ int *le
char entry[],namel[];

{
    *level = 2; /* to bone section level */
    strcpy(entry,"3");
    strcpy(namel,"CL3");
    return;
} /* end of force_now */

/*****
annot_it(entry,level) /* this routine annotates the pictures with description
char entry[]; int level;
{
    static int row[4][3] = {3,3,17,7,10,17,14,16,17,18,0,0};
    static int col[4][3] = {55,14,0,55,14,26,55,14,53,55,0,0};
    static int colr[4][2]={13,15,10,15,14,0,4,15};/*colors for text and boxes
    char string[139],temp[4];
    int use,ccol,crow,count,i,dot,color;

    switch(level) /* set codes for classification */
    {
        case 1:
            temp[0]='1';
            temp[1]='2';
            temp[2]='3';
            temp[3]='4';
            dot=0;
            use = 0;
            break;

        case 2:
            temp[0]='1';
            temp[1]='2';
            temp[2]='3';
            temp[3]='\0';
            dot=0;
            use = 1;
            break;

        case 3:
            temp[0]='A';
            temp[1]='B';

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

        temp[2]='C';
        temp[3]='\0';
        dot=0;
        use = 2;
        break;
case 4:
        temp[0]='1';
        temp[1]='2';
        temp[2]='3';
        temp[3]='\0';
        dot=0;
        use = 2;
        break;
case 5:
        temp[0]='1';
        temp[1]='2';
        temp[2]='3';
        temp[3]='\0';
        dot=1;
        use = 2;
        break;
} /* end switch */

/* find three names and write them out */
/* number 1 */
key_num = 0;
buf_len = sizeof(annotate);
count = 0;
for (count=0;temp[count] != '\0';count++)
{
    if (dot==1) sprintf(key_buf,"%s.%c",entry,temp[count]);
    else sprintf(key_buf,"%s%c",entry,temp[count]);
    for (i=strlen(key_buf);i<10;i++) key_buf[i]=' ';
    status = btrv(B_GETEQ,pos_blk2,&annotate,&buf_len,key_buf,key_num);
    strncpy(string,key_buf,10);
    string[10]='\0';
    ccol = col[count][use] * 8; /* convert to APA */
    crow = row[count][use] * 14;
    /* box color depends upon classification */
    if (temp[count]=='A') color = 1; /* green */
    if (temp[count]=='B') color = 2; /* yellow */
    if (temp[count]=='C') color = 3; /* red */
    if (level > 3 && entry[2]=='A') color = 1; /* green */
    if (level > 3 && entry[2]=='B') color = 2; /* yellow */
    if (level > 3 && entry[2]=='C') color = 3; /* red */
    if (level < 3) color = 0;
    if (level < 3) grrtulc(ccol,crow-16,190,50,colr[color][0]);
    else grrtulc(ccol,crow-16,205,70,colr[color][0]);
    fhatsay(0,string,colr[color][1],ccol,crow);
    strncpy(string,annotate.string,26);
    string[26]='\0';
    ccol = col[count][use] * 8; /* convert to APA */
    crow = (row[count][use]+1) * 14;
    if (status==0) fhatsay(0,string,colr[color][1],ccol,crow);
}

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

        strncpy(string,&annotate.string[26],26);
        string[26]='\0';
        ccol = col[count][use] * 8; /* convert to APA */
        crow = (row[count][use]+2) * 14;
        if (status==0) fhatsay(0,string,colr[color][1],ccol,crow);
        strncpy(string,&annotate.string[52],26);
        string[26]='\0';
        ccol = col[count][use] * 8; /* convert to APA */
        crow = (row[count][use]+3) * 14;
        if (status==0) fhatsay(0,string,colr[color][1],ccol,crow);
    } /* end for */
} /* end of annotate */

/***** re-
move_box(entry) /* this routine removes boxes to show last choice */ char ent
{
    int i,count,ccol,crow;
    static int row[4][3] = {3,3,17,7,10,17,14,16,17,18,0,0};
    static int col[4][3] = {55,14,0,55,14,26,55,14,53,55,0,0};
    static int colr[4][2]={13,15,10,15,14,0,4,15};/*colors for text and boxes
    if (entry == '1') i = 0;
    if (entry == '2') i = 1;
    if (entry == '3') i = 2;
    for (count=0;count<3;count++)
    {
        if (count != i)
        {
            ccol = col[count][2] * 8; /* convert to APA */
            crow = row[count][2] * 14;
            grrtulc(ccol,crow-16,205,70,0);
        } /* end if */
    } /* end for */
} /* end of remove_box */

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/*      COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED      */ #include
<string.h> #include <malloc.h> #include <stdio.h>

/* btrieve definitions */ #define B_OPEN 0 #define B_CLOSE 1 #define B_GETNX
#define B_GETEQ 5 #define B_GETGE 9 #define B_HIGHEST 13 #define B_GETPR 7
#define EOF_ERR 9

/* database parameters */ extern int op; /*operation code*/ extern c
pos_blk1[128]; /*position block*/ extern int buf_len; /* length of above
extern char key_buf[30]; /* key buffer */ extern int key_num; /* key nu
*/ extern int status; /* error status */ extern int
tosteo,tsurg_int,tunreliab,thealing,trecumb; /* tally of percents */
struct
{
    char system[20];/*system code */
    char disease[20];
    char healing[4]; /* % chance of this complication */
    char osteo[4];/* %chance of this complication */
    char recumb[4]; /* % chance of this complication */
    char surg_int[4]; /* % chance of this complication */
    char unreliab[4]; /* % chance of this complication */
    char garb;
} prexst[11];
struct
{
    char system[20];/*system code */
    char descr[40];
    char garb;
} hti_rec[8];

int pre_disease(systq)/* query database for disease questions */ char systq[]
which system is to be queried */ {
    int i,rs,status,rs1;
    char data1[50],data[100];
    int windno,windno2;
    char system[40],string[40],string2[40];
    int numb,lineup;

    int osteo,surg_int,unreliab,healing,recumb; /* the numerical values f
the percents*/
    /* find all the choices, up to 10 maximum */
    windno=0;
    numb=0;
    buf_len = sizeof(prexst);
    key_num = 0;/* system */
    strcpy(system,systq);/* copy for use */
    status = btrv(B_GETGE,pos_blk1,&prexst[numb],&buf_len,system,key_num);
    if (status !=0) fprintf(stderr,"error getting system from database
%i",status);
    strcpy(key_buf,system);/* copy into key buffer for use */
    while (status==0 && strcmp(key_buf,system)==0 && numb < 10)

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

        (
            numb++;
            status = btrv(B_GETNX,pos_blk1,&prexst[numb],&buf_len,key_buf,key_n
        )
    /* create a window area */
    status = 0;
    status= wind_make(windno,375,140,600,numb+1);
    if (status != 0 )fprintf(stderr,"window making error ");
    /* write them to the window */
    strcpy(string,"");
    strncat(string,prexst[0].system,20);
    string[24]=0;
    status=wind_line(windno,string,0,0,13);
    if (status != 0 )fprintf(stderr,"window line error ");
    load_choice();
    for (i=0; i < numb ;i++)
    {
        strncpy(string,prexst[i].disease,20);
        string[20]=0;
        status=wind_line(windno,string,i+1,i+1,15); /* key table loaded autom
cally*/
        if (status != 0 )fprintf(stderr,"window lines error (%i %i
%i)",status,i,windno);
    }
    info_touch(3); /*enable key table response */
    /* loop on touch choices until finished */
    while((rs=info_touch_res(data)) < numb+2 ) /* get response*/
    {
        info_touch(0); /* touch off */
        if (rs !=0) {
            /* accept a point */
            strncpy(string,prexst[rs-1].disease,20);
            string[20]=0;
            status=wind_line(windno,string,rs,0,12);/*dont reload table,change
color*/
            if (status != 0 )fprintf(stderr,"window lines error (%i
%i)",status,windno);
            /* find and write out its information */
            sscanf(prexst[rs-1].healing,"%4i",&healing);
            sscanf(prexst[rs-1].recumb,"%4i",&recumb);
            sscanf(prexst[rs-1].osteo,"%4i",&osteo);
            sscanf(prexst[rs-1].surg_int,"%4i",&surg_int);
            sscanf(prexst[rs-1].unreliab,"%4i",&unreliab);
            windno2 = 1;
            status= wind_make(windno2,280,200,520,5);
            if (status != 0 )fprintf(stderr,"window open error (%i )",status);
            strcpy(string2,string);
            strcat(string2," effects: ");
            string2[29]=0;
            status=wind_line(windno2,string2,0,0,12);
            lineup = 1;
            if (osteo > 0) {
                strcpy(string2,"Osteoporosis: ");
                strncat(string2,prexst[rs-1].osteo,4);
            }
        }
    }

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

        string2[18]='%';
        string2[19]=0;
        status=wind_line(windno2,string2,lineup++,0,12);
    }
    if (surg_int > 0) {
        strcpy(string2,"Surgery concern ");
        strncat(string2,prexst[rs-1].surg_int,4);
        string2[20]='%';
        string2[21]=0;
        status=wind_line(windno2,string2,lineup++,0,12);
    }
    if (unreliab > 0 ){
        strcpy(string2,"Unreliable ");
        strncat(string2,prexst[rs-1].unreliab,4);
        string2[15]='%';
        string2[16]=0;
        status=wind_line(windno2,string2,lineup++,0,12);
    }
    if (recumb > 0 ){
        strcpy(string2,"Recumbancy concern");
        strncat(string2,prexst[rs-1].recumb,4);
        string2[15]='%';
        string2[16]=0;
        status=wind_line(windno2,string2,lineup++,0,12);
    }
    if (healing > 0 ){
        strcpy(string2,"Healing concern ");
        strncat(string2,prexst[rs-1].healing,4);
        string2[15]='%';
        string2[16]=0;
        status=wind_line(windno2,string2,lineup++,0,12);
    }
    if (status != 0 )fprintf(stderr,"window lines error (%i)",status)
/* wait for approval, then write it to two file */
info_touch(3); /*touch on*/
rsl=info_touch_res(data);
info_touch(0);
status=wind_close(windno2);
    if (status != 0 )fprintf(stderr,"window cloase error (%i )",status)
/* one for pcplus with just the relevant complications */
if (rsl==40) wind_line(windno,string,rs,0,15);/*color it undone */
    else {
        thealing = tally(thealing,healing);
        trecumb = tally(trecumb,recumb);
        tosteo = tally(tosteo,osteo);
        tunreliab = tally(tunreliab,unreliab);
        tsurg_int = tally(tsurg_int,surg_int);
    }
/* fix this - add saving of diseases chosen */
} /*end zero if*/
info_touch(3); /*touch on */
} /* end loop */
/* put back window */
status = wind_close(windno);

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

    if (status != 0) fprintf(stderr, "window closing error (%i
%i)", status, windno);
    return(0);
} /*end of pre_disease */

/*****

int tally(previous,new) /* this function tallys CF's , as does PC PLUS */
/* the return value is the tallied CF */ int previous,new;

{
    int tal;

    if (new == 0) return(previous);
    if (previous >= 0 && new >= 0)
    {
        tal = previous + (( new * (100 - previous)) +50)/100.;
    }
    else {
        /* this should not occur */
        fprintf(stderr, "problem calculating CF %i %i",previous,new);
        exit(1);
    }
    return(tal);
}
/* end of tally */
*****/

int get_applic(concern)/* query database by concern */ int concern; /* which
concern is to be queried */ {
    int i,rs,status,rsl;
    char data1[50],data[100];
    int windno;
    char system[40],string[40];
    int numb,lineup;

    int healing,recumb,osteo,surg_int,unreliab;      /* the numerical values f
the percents*/

    /* open database for preexisting illnesses*/
    status = btrv(B_OPEN,pos_blk1,&prexst[0],&buf_len,"preexist.dat",0);
    if (status!=0) printf("database problem %i,status");

    /* find all those with this concern, up to 10 maximum */
    windno=0;
    numb=0;
    buf_len = sizeof(prexst);
    key_num = 0; /* system */
    status = btrv(B_HIGHEST,pos_blk1,&prexst[numb],&buf_len,system,key_nu
    if (status !=0) fprintf(stderr,"error getting system from database
%i",status);
    strcpy(key_buf,system);/* copy into key buffer for use */
    while (status==0 && numb < 10)

```


COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

    {
        /* translate to numbers */
        sscanf(prexst[numb].healing,"%4i",&healing);
        sscanf(prexst[numb].recumb,"%4i",&recumb);
        sscanf(prexst[numb].osteo,"%4i",&osteo);
        sscanf(prexst[numb].surg_int,"%4i",&surg_int);
        sscanf(prexst[numb].unreliab,"%4i",&unreliab);
        /* only keep it if it involves the present concern */
        if (concern == 1 && surg_int > 0) numb++;
        if (concern == 2 && healing > 0) numb++;
        if (concern == 3 && recumb > 0) numb++;
        if (concern == 4 && unreliab > 0) numb++;
        if (concern == 5 && osteo > 0) numb++;
        status = btrv(B_GETPR,pos_blk1,&prexst[numb],&buf_len,key_buf,key_n
    ) /* end while */

    /* create a window area */
    status = 0;
    status= wind_make(windno,75,10,300,numb+1);
    if (status != 0 ) {
        fprintf(stderr,"window making error get_,windn:%i numb %i",windno,num
        mayexit();
    }
    /* write them to the window */
    status=wind_line(windno," Example problems:",0,0,13);
    if (status != 0 ) fprintf(stderr,"window line error ");
    load_choice();
    for (i=0; i < numb ;i++)
    {
        strncpy(string,prexst[i].disease,20);
        string[20]=0;
        status=wind_line(windno,string,i+1,0,15); /* no keys*/
        if (status != 0 ) fprintf(stderr,"window lines error (%i %i
%i)",status,i,windno);
    }
    return(0);
} /*end of get_applic */

/*****

int get_descr(system)/* query database by system for hti */ char system[]; /*
which system is to be queried */ {
    int i,rs,status,rs1;
    char datal[50],data[100];
    int windno;
    char string[41],syssav[20];
    int numb,lineup;

    /* open database for hti*/
    status = btrv(B_OPEN,pos_blk1,&hti_rec[0],&buf_len,"htrauma.dat",0);
    if (status!=0) printf("database problem %i,status");

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/* find all those with this system, up to 7 maximum */
windno=0;
numb=0;
buf_len = sizeof(hti_rec);
strcpy(key_buf,""); /* clear some */
strncpy(key_buf,system,4);
key_num = 0; /* system */
status = btrv(B_GETGE,pos_blk1,&hti_rec[numb],&buf_len,key_buf,key_num);
if (status !=0) fprintf(stderr,"error getting system from database
%i",status);
strncpy(syssav,key_buf,20);
while (status==0 && numb < 8 && (strcmp(syssav,key_buf,20)==0) )
{
    numb++;
    status =
btrv(B_GETNX,pos_blk1,&hti_rec[numb],&buf_len,key_buf,key_num);
} /* end while */
numb--; /* disregard the last one */
/* create a window area */
status = 0;
status= wind_make(windno,75,10,400,numb+1);
if (status != 0 ) {
    fprintf(stderr,"window making error get_,windn:%i numb %i",windno,numb);
    mayexit();
}
/* write them to the window */
status=wind_line(windno," Index descriptions:",0,0,13);
if (status != 0 ) fprintf(stderr,"window line error ");
load_choice();
for (i=0; i < numb ;i++)
{
    strncpy(string,hti_rec[i].descr,40);
    string[40]=0;
    status=wind_line(windno,string,i+1,0,15); /* no keys*/
    if (status != 0 ) fprintf(stderr,"window lines error (%i %i
%i)",status,i,windno);
}
return(0);
} /*end of get_descr */

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/*      COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED      */ #include
<dos.h> /* this contains subroutines for the system disease info gathering
/* Bob Friedman July 1987 */

/* btrieve definitions */ #define B_OPEN 0 #define B_CLOSE 1 #define B_GETNX
#define B_GETEQ 5 #define B_GETGE 9 #define EOF_ERR 9 /* database parameters
extern int op; /*operation code*/ extern char pos_blk1[128]; /*posi
block*/ extern char pos_blk2[128]; extern int buf_len; /* length of above
extern char key_buf[30]; /* key buffer */ extern int key_num; /* key nu
*/ extern int status; /* error status */ extern char entry[30]; /* key
*/ extern struct
{
    char system[20]; /*system code */
    char disease[20];
    char healing[4]; /* % chance of this complication */
    char osteo[4]; /* %chance of this complication */
    char recumb[4]; /* % chance of this complication */
    char surg_int[4]; /* % chance of this complication */
    char unreliab[4]; /* % chance of this complication */
    char garb;
} prexst[11];
int tosteo,tsurg_int,tunreliab,thealing,trecumb; /* tally of percents */

/*****/ int con-
cern() /* gather concern statistics */
{
    static char datal[30]={
        1,2,2,12,26,2,3,25,12,48,3,4,48,12,72,4,12,3,21,22,
        5,12,25,21,46,6,13,49,22,70};
    static char data2[30]={
        1,11,4,16,31,2,12,32,16,57,3,12,59,16,80,4,15,6,19,31,
        5,16,34,19,57,6,16,57,19,78};
    char data[120],name[40];
    int rs,status,rsl,value;
    thealing = trecumb = tosteo = tsurg_int = tunreliab = 0; /*initially*/
    load_choice();
    /*preexisting systemic */

    /*get from concern choices */
    load_choice();
    strncpy(data,datal,30);
    info_load_key(30,data); /*set up key table*/
    info_touch(3); /*touch on */
    while((rs=info_touch_res(data)) < 7) /* get response*/
    {
        info_touch(0);
        if (rs == 6) { /* specific disease entry */
            part_halo_draw("c:\\clinical\\body.pic",4);
            specif();
        }
        if (rs > 0 && rs < 6) {
            part_halo_draw("c:\\clinical\\enter.pic",4); /*has mild,moderate.

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

    get_applic(rs); /* find applicable diseases */
    /* get response - 1 to 6 */
    load_choice();
    strncpy(data,data2,30);
    info_load_key(30,data);
    info_touch(3); /* touch on */
    while((rs1=info_touch_res(data)) < 1 ); /* short loop get respons
    info_touch(0);
    value = (rs-1)*20.;
    switch(rs) {
    /* which concern */
    case 1: /*surgery */
        tsurg_int = tally(tsurg_int,value);
        break;
    case 2: /* healing */
        thealing = tally(thealing,value);
        break;
    case 3: /*recumbancy */
        trecumb = tally(trecumb,value);
        break;
    case 4: /*patient reliability */
        tunreliab = tally(tunreliab,value);
        break;
    case 5: /* bone quality */
        tosteo = tally(tosteo,value);
        break;
    } /* end switch */
    } /* end if */
    status = wind_close(0); /* must close window used in get_applic */
    if (status !=0) printf("error closing window ");
    part_halo_draw("c:\\clinical\\concern.pic",4);
    load_choice();
    strncpy(data,data2,30);
    info_load_key(30,data); /*set up key table*/
    info_touch(3); /* touch on */
    } /* end while*/
/* output results */
yesno_out("yes",3,thealing,"healing");
yesno_out("yes",3,tosteo,"osteoporotic");
yesno_out("yes",3,trecumb,"recumbancy");
yesno_out("yes",3,tsurg_int,"surgery intolerability");
yesno_out("yes",3,tunreliab,"unreliable patient ");
if (rs == 40) return(1);
if (rs == 41) return(0);
if (rs == 42) mayexit();
return;
} /* end of body */

/*****/ int spec
/* gather body stats */
{
    static char data2[55]={
        8,10,39,13,57,9,13,39,16,57,10,16,39,19,57,11,19,39,24,57,
        12,1,57,3,80,13,3,57,7,80,14,7,57,10,80,15,10,57,13,80,

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

        16,13,57,16,80,17,16,57,19,80,18,19,57,23,80);
static char data3[20]={
    4,2,2,5,19,5,5,2,8,19,6,8,2,11,19,7,11,2,16,19};
char data[120],name[40];
int rs,status;
static int ulr = {0};
static int ulc = {136};
static int lrr = {295};
static int lrc = {324};
static char files[11][12]={"CARD","DIGEST","ENDO","INTEG","LYMPH",
    "MUSC","NERVOUS","REPROD","RESPIRA","SKEL","URINARY"};
load_choice();
/*preexisting systemic */
setegapr(1,61); /* change color to red */
/* open database for preexisting illnesses*/
status = btrv(B_OPEN,pos_blk1,prexst,&buf_len,"preexist.dat",0);
if (status!=0) printf("database problem %i,status");

/*get from system choices */
load_choice();
strncpy(data,data2,55);
info_load_key(55,data);/*set up key table*/
info_touch(3); /*touch on */
while((rs=info_touch_res(data)) < 19) /* get response*/
{
    if (rs > 7) {
        info_touch(0); /* touch off */
        clipsize(ulc,ulr,lrc,lrr);
        strcpy(name,"c:\\clinical\\");
        strcat(name,files[rs-8]);
        strcat(name,".pic");
        part_halo_draw(name,1);
        status = pre_disease(&files[rs-8][0]);
        load_choice();
        strncpy(data,data2,55);
        info_load_key(55,data);/*set up key table*/
        info_touch(3); /*touch on */
    }
} /* end while*/
/* close database */
status = btrv(B_CLOSE,pos_blk1,prexst,&buf_len,key_buf,0);
if (status != 0)
{
    printf("Error closing file . Status = %d",status);
    exit(0);
}
if (rs == 40) return(1);
if (rs == 41) return(0);
if (rs == 42) mayexit();
return;
} /* end of specif */

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/*      COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED      */ #include
<dos.h> #include <search.h> /* btrieve definitions */ #define B_OPEN 0 #defin
B_CLOSE 1 #define B_GETNX 6 #define B_GETEQ 5 #define B_GETGE 9 #define EOF_E
/* database parameters */ extern int op; /*operation code*/ extern c
pos_blk1[128]; /*position block*/ extern char pos_blk2[128]; extern int
buf_len; /* length of above */ extern char key_buf[30]; /* key buffer */
extern int key_num; /* key number */ extern int status; /* error st
*/ extern char entry[30]; /* key copy */ static struct
{
    int extr; /*values of hti by system */
    int nerv;
    int resp;
    int abdm;
    int card;
    int skin;
    int comp;
    int iss;
} hti_det={0,0,0,0,0,0,0,0};

/*****/ int hti(
gather health trauma index */
{
    static char datal[35]={
        1,2,9,14,23,2,9,2,22,12,3,10,24,21,37,4,3,36,15,47,
        5,10,47,21,59,6,2,58,14,69,7,10,71,20,78};
    static char data2[30]={
        1,11,4,16,31,2,12,32,16,57,3,12,59,16,80,4,15,6,19,31,
        5,16,34,19,57,6,16,57,19,78};
    static char system[7][5]={"EXTR","NERV","RESP","ABDO","CARD",
    "SKIN","COMP"};
    char data[120],name[40];
    int rs,status,rs1,value;

    /*get from system choices */
    load_choice();
    strncpy(data,datal,35);
    info_load_key(35,data); /*set up key table*/
    info_touch(3); /*touch on */
    while((rs=info_touch_res(data)) < 8) /* get response*/
    {
        info_touch(0);
        if (rs > 0 && rs < 8) {
            part_halo_draw("c:\\clinical\\enter.pic",4); /*has mild,moderate.
            get_descr(system[rs-1]); /* find descriptions of symptoms */
            /* get response - 1 to 6 */
            load_choice();
            strncpy(data,data2,30);
            info_load_key(30,data);
            info_touch(3); /* touch on */
            while((rs1=info_touch_res(data)) < 1 ); /* short loop get respons
            info_touch(0);
            value = rs1-1;
            switch(rs) {

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/* which system */
case 1: /*extremities */
    hti_det.extr = value;
    break;
case 2: /* nervous */
    hti_det.nerv = value;
    break;
case 3: /* respiratory */
    hti_det.resp = value;
    break;
case 4: /* abdominal */
    hti_det.abdm = value;
    break;
case 5: /* cardiovascular */
    hti_det.card = value;
    break;
case 6: /* skin and subcutaneous */
    hti_det.skin = value;
    break;
case 7: /* complications */
    hti_det.comp = value;
    break;
} /* end switch */
} /* end if */
status = wind_close(0); /* must close window used in get_descr */
if (status !=0) printf("error closing window ");
part_halo_draw("c:\\clinical\\hti.pic",4);
load_choice();
strncpy(data,data1,35);
info_load_key(35,data);/*set up key table*/
info_touch(3); /* touch on */
} /* end while*/
find_iss(); /* determine the injury severity score */
/* output results */
numb_out(hti_det.extr,100,"extremity ");
numb_out(hti_det.nerv,100,"nervous ");
numb_out(hti_det.resp,100,"respiratory ");
numb_out(hti_det.abdm,100,"abdominal ");
numb_out(hti_det.card,100,"cardiovascular ");
numb_out(hti_det.skin,100,"skin ");
numb_out(hti_det.comp,100,"complications ");
numb_out(hti_det.iss,100,"injury severity score");
if (rs == 40) return(1);
if (rs == 41) return(0);
if (rs == 42) mayexit();
return;
} /* end of body */
/*****/ int find_i
/* find the injury severity score defined as the sum */ /* of the three highe
hti values */

{
    int i,sort[7];
    /* copy numbers */ ;

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

for (i=0;i < 7; i++) {
    sort[i] = *(&hti_det.extr + i);
}
/* sort the array */
ssort(sort,7);
/* now sum the squares */
hti_det.iss = sort[0]*sort[0] + sort[1]*sort[1] + sort[2]*sort[2];
return;
}/* end of find_iss*/
/***** int
ssort(list,numb) /* does numerical sorting */ int list[],numb; {
    int i,j;
    int temp;

    for (i=1;i<numb;i=i+1)
    {
        temp=list[i];
        j = i-1;
        while( (j >=0) && (temp > list[j]))
        {
            list[j+1] = list[j];
            j=j-1;
        }
        list[j+1] = temp;
    } /* end of ssort */
}

```


COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/*      COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED      */ #include
<stdio.h> #include <string.h> #include <ctype.h>

/***** program to determine final summary for aoca *****/ /* Bob Friedma
June 1987 */

/* btrieve definitions */ #define B_OPEN 0 #define B_CLOSE 1 #define B_GETNX
#define B_GETPR 7 #define B_GETEQ 5 #define B_GETGE 9 #define EOF_ERR 9 #defi
B_UPDATE 3 #define B_HIGHEST 13

static int op;          /*operation code*/ static char pos_blk1[128]; /*posi
block*/ static int buf_len; /* length of above */ static char key_buf[30
/* key buffer */ static int key_num; /* key number */ static int certain
/* certainty factor read */ static int status; /* status value returned
static struct
{
    char high[30]; /*treatment of highest level */
    char med[30]; /* treatment of medium level */
    char low[30]; /* treatment of low level */
    char lowest[30]; /* treatment of lowest level */
    short int cerfac; /* holds numbers for ordering */
    char file[5]; /* file name for implant picture */
    char start[5]; /* start of movie */
    char stop[5]; /* end of movie */
    char empty[1]; /* empty space */
} treat_rec;

main() {
    int status,numb;
    char i,data[10];
    FILE *tempf; /* temporary file for data passing */

    if ( ( tempf = fopen("temper.txt","r")) == NULL) {
        fprintf(stderr,"%s couldnt open file 'temper.txt'");
        exit(1);
    }

    /* open database for procedure classification */
    status = btrv(B_OPEN,pos_blk1,&treat_rec,&buf_len,"procedur.tmp",0);
    if (status != 0)
    {
        printf("Error opening file . Status = %d",status);
        exit(0);
    }

    info_stop(); /*clear problems info window*/
    info_start(); /*initialize info window*/
    easyinit(); /*graphics initialization */
    numb=1;
    status=fontld(numb,"c:\\clinical\\ROMAN3I.esi");
    if (status!=0) printf("error in font setting %i %i",status,numb);
    update(tempf); /* update the database to hold CF values obtained*/
    status = btrv(B_CLOSE,pos_blk1,&treat_rec,&buf_len,key_buf,0);
    if (status != 0)

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

        printf("Error closing file . Status = %d",status);
        exit(0);
    }
    befacht(); /*results and the before and after picture */
    piecht(); /* display an appropriate pie chart */
    fcloseall(); /* close all files */
    printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
    printf("\rhit return to continue:");
    while(kbhit() == 0);/*wait*/
    initgraf(3,0,0);
    info_stop();
    exit();
} /* end of main program for initial data */
/***** int up-
date(tempf)
FILE *tempf; {

static char scopy[30];

getone(tempf,key_buf,&certain); /* get a program output */
while (certain != -500) /* do this for each data item */
{
    /* now find this in the procedure classification database */
    buf_len = sizeof(treat_rec);
    key_num = 0;/* first level code */
    status = btrv(B_GETEQ,pos_blk1,&treat_rec,&buf_len,key_buf,key_num);
    while (status !=0) /*look to lower levels */
    {
        key_num++;
        if (key_num > 3)
        {
            printf("Error classification procedure mismatch. Status
%d",status);
            exit (0);
        }
        status =
btrv(B_GETEQ,pos_blk1,&treat_rec,&buf_len,key_buf,key_num);
    }

    /* now update with certainty factor */
    treat_rec.cerfac += certain*(key_num+4); /*weighting */
    status = btrv(B_UPDATE,pos_blk1,&treat_rec,&buf_len,key_buf,key_num);
    if (status !=0 )
    {
        printf("Error in updating certainties. Status %d",status);
        exit (0);
    }

    /* now update any others on the same level */
    strcpy(scopy,key_buf); /* get a copy of the key buffer */
    while (strcmp(key_buf,scopy)==0 && status == 0) /* the same string */
    {
        /* see if there exists a next one */

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

        status = btrv(B_GETNX,pos_blk1,&treat_rec,&buf_len,key_buf,key_n
        if (strcmp(key_buf,scopy)==0 && status == 0) /* no error and the
same string */
        {
            treat_rec.cerfac += certain*(key_num+1); /*weighting */
            status =
btrv(B_UPDATE,pos_blk1,&treat_rec,&buf_len,key_buf,key_num);
            if (status !=0 )
            {
                printf("Error in updating certainties. Status %d",status);
                exit (0);
            } /* end status if */
        } /* end compare if */
    } /* end compare while */
    getone(tempf,key_buf,&certain);
} /*end while*/
return(0); } /* end of  update */
/*****/

int getone(tempf,string,numb) /* read in characters 1 at a time */
    /* to place together a parameter and certainty */

FILE *tempf; char string[]; short int *numb;

{
    char check;
    static test[2]={' ','\n'};
    int count,found;
    found = 0;
    *numb = -500; /* initialize this */
    /* fill with blanks */
    for (count = 0; count< 30 ;count++)
        string[count]=' ';
    /* skip any initial parentheses ,spaces or quotes */
    while (isalnum(check=fgetc(tempf)) == 0)
    {
        if (check =='\n') found = 1; /* note if quote appears */
        /* check for end of file */
        if (feof(tempf) != 0) return;
    }
    /* read in until test character occurs */
    for (count = 0; check != test[found] ; count++)
    {
        if (count > 255)
        {
            printf("error with rd file ");
            exit(1);
        }
        string[count] = check;
        check = fgetc(tempf);
    }
    /* read in a number */
    fscanf(tempf,"%i",numb);
    return(0);
}

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

} /* end getone */

/*****
data_out(string,numb) /* this routine writes out the string, without trailing
                        /* to a file in a format PCPLUS will recognize */
char string[]; int numb;
{
    int i;

    /* determine how many trailing blanks there are */
    while (string[numb-1] == 32 )
        numb--;

    for (i=0; i < numb; i++)
        printf("%c",string[i]); /* print out non-blank character*/
    return(0); } /* end of data_out */

/*****
piecht() /* display an appropriate pie chart */ {
    int i,x,y,r,s,e,c,off,args;
    static int pnts[10]={10,33,12,25,40,9,17,54,25,1010};
    initgraf(16,2,4);
    /* display the chart showing femoral fracture complications */
    x=320; /*center of chart */
    y=175;
    r = 150; /*radius */
    c = 12; /*border color*/
    off = 10;
    e = 512; /*explode slice 10 */
    s = 0;
    args=10; /*10 pieces of data */
    fhatsay(1,"Complications",3,50,30);
    fhatsay(0,"while in the hospital",3,5,45);
    fhatsay(0,"of femoral fracture patients",3,5,55);
    fhatsay(0,"treated with AO techniques",3,5,65);
    fhatsay(0,"Switzerland and Germany",3,5,75);
    fhatsay(0,"1973-1976 n=1275",3,5,85);
    i=hsset(0,0," Shock",10,0,1);
    if (i!=0) printf("error in graph setting1 %i",i);
    i=hsset(1,0," Respiratory",10,0,11);
    if (i!=0) printf("error in graph setting2 %i",i);
    i=hsset(2,0," Pulmonary embolism",10,0,11);
    if (i!=0) printf("error in graph setting3 %i",i);
    i=hsset(3,0," Fat embolism",10,0,11);
    if (i!=0) printf("error in graph setting4 %i",i);
    i=hsset(4,0," Cardio-vascular",10,0,11);
    if (i!=0) printf("error in graph setting5 %i",i);
    i=hsset(5,0," Thrombosis",10,0,11);
    if (i!=0) printf("error in graph setting6 %i",i);
    i=hsset(6,0,"CNS",10,0,11);

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```
if (i!=0) printf("error in graph setting7 %i",i);
i=hsset(7,0,"Others",10,0,11);
if (i!=0) printf("error in graph settings8 %i",i);
i=hsset(8,0,"Paresis",10,0,11);
if (i!=0) printf("error in graph settings9 %i",i);
i=hsset(9,0,"No Complications",10,0,11);
if (i!=0) printf("error in graph setting10 %i",i);
i=hspie(x,y,r,s,e,c,off,args,pnts);
if (i!=0) printf("error in graph drawing11 %i",i);
return(0);
} /*end piecht*/
```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/*      COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED      */ #include
<stdio.h> #include <stdlib.h>

movie(start,stop)          /* this is to display movie results */ int start,st
{

int row,col,color,kern,mode,assign; int x,i,j; short int count,frame; char on
char colr; int choice,pause; short int zero,same; char data[10];

/* some intialization */
    frame = start; /*the beginning */
    zero = 0; /* for no information */
    same = 0xFFFF; /* for same frame */
    pause = 0; /* not paused */
    info_init_video(); /*start player*/
    info_enable_audio(); /*enable audio channels */
    info_display(1); /*show the frame number */ /* draw some graphics and ena
key areas */

    /* viewing area */
    initgraf(14,2,0); /* mode of 640 X 200 16 colors*/
        color = 4;
        colr = 4;
        info_trans(colr); /*set transparent */
        info_video_control(); /*enable transparency*/
        grrtulc(0,0,640,200,color);
    info_load_key(0,data); /*initialize keys */
    /* skip foward */
        keyit(1,"Start",500,25);
    /* foward 2*/
        keyit(2,"Pause",500,75);
    /* end 3*/
        keyit(3,"Backwards",500,110);
    /* backwards4 */
        keyit(4,"Continue",500,140);
    /* skip backwards5 */
        keyit(5,"End",500,180);

    /* get a screen response */

    onoff = 3; /* for key interface */
    info_touch(onoff); /*activate screen */ /* while not end */
    while((choice=info_touch_res(data)) < 5)
    {
        switch (choice) {
            case 1 :
                /* start */
                if (pause==0 ) {
                    info_from_to(zero,zero); /* pauses player */
                    wait(2);
                }
                info_from_to(start,stop);

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

        pause = 0;
        break;
    case 2 :
        /* pause */
        info_from_to(zero,zero); /* pauses player */
        pause = 1;
        break;
    case 3 :
        /* backwards */
        if (pause==0) {
            info_from_to(zero,zero); /* pauses player */
            wait(2);
        }
        info_from_to(same,start);
        pause = 0;
        break;
    case 4 :
        /*continue*/
        if (pause==0) {
            info_from_to(zero,zero); /* pauses player */
            wait(2);
        }
        info_from_to(same,stop);
        pause=0;
        break;
    } /* end switch */
} /* end while */ /* for end */ info_exit();/*stop video player */ /* bac
normal mode */

easyinit(); /* redo graphics */
return;
}

/*****
keyit(keyno,stng,col,row) /*this draws a string at a point of APA sets key
int keyno; char stng[]; int col,row;
{
    short int count;
    char onoff,data[10];
    int colr,length,font;

    /* draw string at this position */
    colr = 3;
    font = 0;
    fhatsay(font,stng,colr,col,row);
    /* load the key for this boundary */
    data[0] = keyno;
    data[1] = row/8-1;
    data[2] = col/8 -1;
    data[3] = row/8+2;
    data[4] = col/8 + 6;
    info_load_key(5,data); } /* end of keyit*/

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/*      COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED      */ #include
<stdio.h> #include <dos.h> #include <stdlib.h> #include <memory.h> #include
<conio.h>

/* btrieve definitions */ #define B_OPEN 0 #define B_CLOSE 1 #define B_GETNX
#define B_GETPR 7 #define B_GETEQ 5 #define B_GETGE 9 #define EOF_ERR 9 #defi
B_UPDATE 3 #define B_HIGHEST 13

#define TRUE 1 #define FALSE 0

static int op;          /*operation code*/ static char pos_blk1[128]; /*posi
block*/ static int buf_len; /* length of above */ static char key_buf[30
/* key buffer */ static int key_num; /* key number */ static int certain
/* certainty factor read */ static int status; /* status value returned
static struct
{
    char high[30]; /*treatment of highest level */
    char med[30]; /* treatment of medium level */
    char low[30]; /* treatment of low level */
    char lowest[30]; /* treatment of lowest level */
    short int cerfac; /* holds numbers for ordering */
    char file[5]; /* file name for implant picture */
    char start[5]; /* start of movie */
    char stop[5]; /* end of movie */
    char empty[1]; /* empty space */
} treat_rec[5];

int beaft() /* this function draws the before-after picture of the implant
{ unsigned char file[40],file2[40],data[50]; FILE *stream,*tempf; int
s,count,rs,redraw_needed,finished; short int start,stop; /* for movie frames
static int x[3]={12,240,445}; static int y[3]={22,22,22}; char
class[10],name[40],cutfile[40]; int numb,bef,aft,sum;

/* open database for procedure classification */ status =
btrv(B_OPEN,pos_blk1,treat_rec,&buf_len,"procedur.tmp",0); if (status != 0)
{
    printf("Error opening file . Status = %d",status);
    exit(0);
}

/* open the file to obtain the classification number */ if ( ( tempf =
fopen("trauma.rd","r")) == NULL) { fprintf(stderr,"%s couldnt open file 'trau
ma.rd'");
    exit(1);
} /* get the classification */ getone(tempf,class,&numb); fclose(tempf); /
first display the picture of the classification */
strcpy(name,"c:\\clinical\\cl"); strcat(name,class,4); strcat(name,".pic");
determine where picture of interest is */ if (class[5] == '1') bef = 1; if
(class[5] == '2') bef = 2; if (class[5] == '3') bef = 3; /* determine locatio
for other boxes */ switch (bef) {

```


COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

case 1:
    sum = 2;
    aft = 3;
    break;
case 2:
    sum = 1;
    aft = 3;
    break;
case 3:
    sum = 1;
    aft = 2;
    break; } /*****/ redraw_needed = TRUE; finished = FALSE; while (re-
draw needed & !finished)
{
    set_choice();
    count = 0; /* no choices yet */
    /* draw the picture */
    status=part_halo_draw(name,4);
    /* over 1 draw the cut file picture */
    if (class[1] == '1')
cut_halo_draw("c:\\clinical\\c31.cut",x[aft-1],y[aft-1]);
    if (class[1] == '2')
cut_halo_draw("c:\\clinical\\c32.cut",x[aft-1],y[aft-1]);
    if (class[1] == '3')
cut_halo_draw("c:\\clinical\\c33.cut",x[aft-1],y[aft-1]);
    /* over one draw a summary */
    final_summ(x[sum-1],y[sum-1],class);

    /* display the choices */
    key_num = 4; /* set to certainty key */
    buf_len = sizeof(treat_rec);
    status = btrv(B_HIGHEST,pos_blk1,&treat_rec[0],&buf_len,key_buf,key_num);
    while (treat_rec[count].cerfac > (8224 + 400) && count < 4) /* the ones
reasonably suggested*/
    {
        explain(count);
        count++;
        status =
btrv(B_GETPR,pos_blk1,&treat_rec[count],&buf_len,key_buf,key_num);
    }
    if (status !=0 )
    {
        printf("Error in finding highest. Status %d",status);
        exit (0);
    }
    /* read a touch choice */
    info_touch(3); /*enable key table response */
    redraw_needed = FALSE;
    rs=info_touch_res(data);
    while(!redraw_needed && !finished)
    {
        info_touch(0);
        if (rs > 5) finished=TRUE;
        if (rs >0 && rs < 5) { /* check if additional pictures are availabl

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

        if (treat_rec[rs-1].file[0] != ' ') (
            /* aha found an implant */

sprintf(cutfile,"c:\\clinical\\%.5s.cut",treat_rec[rs-1].file);
        s = cut_halo_draw(cutfile,x[aft-1],y[aft-1]+15);
        s = cut_halo_draw(cutfile,x[bef-1],y[bef-1]+15);
        redraw_needed = TRUE; /* will need to redraw */
    )
    /* find movie frame numbers */
    sscanf(treat_rec[rs-1].start,"%5i",&start);
    sscanf(treat_rec[rs-1].stop,"%5i",&stop);
    if (start!=0) {
        /* aha a movie */
        if (redraw_needed) wait(8); /* wait 8 seconds */
        movie(start,stop);
        redraw_needed = TRUE; /* will need to redraw */
    }

    } /* end rs if */
    info_touch(3);
    if (!redraw_needed) rs=info_touch_res(data);
    } /* end while */
} /* end redraw section */
status = btrv(B_CLOSE,pos_blk1,treat_rec,&buf_len,key_buf,0);
if (status != 0)
{
    printf("Error closing file . Status = %d",status);
    exit(0);
}

if (rs == 40) return(1); /*warning - recursion */
if (rs == 41) return(0);
if (rs == 42) mayexit();
return(count); /* return the number found */ } /******
end of beaft() *****/ int explain(count) /* print ou
conclusions in appropriate places */ int count; {
    char data[100];
    int length,width;
    static int x[4]={50,50,400,400};
    static int y[4]={150,225,150,225};

    width = 200;
    length = 57; /* make box */
    grrtulc(x[count],y[count],width,length,13); /* set up touch area */
    data[0] = count+1;
    data[1] = y[count]/14.;
    data[2] = x[count]/8.;
    data[3] = (y[count]+length)/14;
    data[4] = (x[count]+width)/8;

    info_load_key(5,data); /* set up key table*/ /* print out results */
    sprintf(data," %.30s",treat_rec[count].high);
    fhatsay(0,data,15,x[count]+14,y[count]+7);
    sprintf(data," %.30s",treat_rec[count].med);
    fhatsay(0,data,15,x[count]+14,y[count]+21);

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

sprintf(data," %.30s",treat_rec[count].lowest);
fhatsay(0,data,15,x[count]+14,y[count]+35);
sprintf(data," %.30s",treat_rec[count].low);
fhatsay(0,data,15,x[count]+14,y[count]+49);
return(0);
} /*end explain */
/*****
final_summ(x,y,class) /* routine to draw a final summary of the patient *
int x,y; char class[10];

{
    FILE *tempf;
    char string[40],string1[80];
    int numb;

/* make a box */
    grrtalc(x,y,200,100,4); /* classification */
    /* open the file to obtain the class */
    if ( ( tempf = fopen("trauma.rd","r")) == NULL) {
        fprintf(stderr,"%s couldnt open file 'trauma.rd'");
        exit(1);
    }
    /* get the age */
    getone(tempf,string,&numb);
    fclose(tempf);
    sprintf(string1,"Classification %.10s",string);
    fhatsay(0,string1,15,x+14,y+14); /* age */
    /* open the file to obtain the patient age */
    if ( ( tempf = fopen("patient.rd","r")) == NULL) {
        fprintf(stderr,"%s couldnt open file 'patient.rd'");
        exit(1);
    }
    /* get the age */
    getone(tempf,string,&numb);
    fclose(tempf);
    sprintf(string1,"Patient Age %.3s",string);
    fhatsay(0,string1,15,x+14,y+28); /* open tissue */
    /* open the file to obtain the open grade */
    if ( ( tempf = fopen("tissue.rd","r")) == NULL) {
        fprintf(stderr,"%s couldnt open file 'tissue.rd'");
        exit(1);
    }
    /* get the openness */
    getone(tempf,string,&numb);
    getone(tempf,string,&numb);
    getone(tempf,string,&numb);
    getone(tempf,string,&numb);
    getone(tempf,string,&numb);
    fclose(tempf);
    sprintf(string1,"Openness %.1s",string);
    fhatsay(0,string1,15,x+14,y+42); /* concerns */
    /* open the file to obtain the concerns */
    if ( ( tempf = fopen("disease.rd","r")) == NULL) {
        fprintf(stderr,"%s couldnt open file 'disease.rd'");

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

    exit(1);
}
/* get the osteo number */
getone(tempf,string,&numb);
sprintf(string1,"Osteoporosis %i%%",numb);
fhatsay(0,string1,15,x+14,y+56);
/* get the surgery number */
getone(tempf,string,&numb);
sprintf(string1,"Surgery %i%%",numb);
fhatsay(0,string1,15,x+14,y+68);
/* get the Reliability number */
getone(tempf,string,&numb);
sprintf(string1,"Reliability %i%%",numb);
fhatsay(0,string1,15,x+14,y+82);
fclose(tempf);
return; }/* end of final sum */
/*****/ wait(seco
/* routine to wait a number of seconds */ int seconds;

{
    long ltime,wtime;

    time(&wtime); /* current time */
    while((ltime-wtime) < seconds)
    {
        time(&ltime);
    }
    return;
} /* end of wait */

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/*      COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED      */
/*****
 * Library routines for using the IBM InfoWindow.
 * *****/

#include <dos.h>

/* Definitons
/* Destinations
#define ERROR      0x00 #define SYSTEM      0x01 #define VIDEO      0x02 #define
ROMSPEECH 0x03 #define EXTSPEECH 0x04 #define TOUCH      0x05 #define VDP1
0x06 #define VDP2      0x07 #define TIME      0x08

/* Commands
#define START      0x01 #define STOP      0x00 #define VIDEO_CNTRL 0x01
#define TRANSPARENT 0x03 #define ROM_SPEAK 0x01 #define TOUCH_CNTRL 0x01
#define KEY_TABLE  0x02 #define TOUCH_RESP 0x04 #define SOUND_BEEP 0x05
#define VDP_INIT   0x01 #define VDP_EXIT  0x12 #define VDP_AUDIO  0x07
#define MATCH_FRAME 0x09 #define STILL_FRAME 0x02 #define PLAY      0x04
#define FROM_TO     0x03 #define WAKEUP    0x03 #define READ_FRAME 0x0A
#define DISPLAY     0x06

#define matchframe_response 3 #define outside_touch 4 #define resp_invalid 5
#define TRUE 1; #define FALSE 0; /* GLOBAL DECLARATIONS
*/

/* Command buffer for sends and receives
struct databuf {
    short int count;
    char id;
    char destination;
    char command;
    unsigned char data [247];
};

struct cntbuf {
    short int kind;
    unsigned short int off_data;
    unsigned short int seg_data;
    unsigned short int off_global;
    unsigned short int seg_global;
};

/* Global variables defined at the external level
*/ struct {
    short int sndst;
    short int recst;
    short int sta0;
    short int ver;
    short int dcode;
    short int pcode;
} avi;

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

extern int WAKE_UP_ROUTINE ();

/* Define work space for calculating address of wake up routine.
union
{
    long int address;
    char addr_array [4];
} wakeup_address;

/*****
info_start() /*
*
* Send Start command and wait to receive Start Response.
*
{
    short int count;
    char id,destination,command,data[1];

    count = 5;
    id = 1;
    destination = SYSTEM;
    command = START;

    info_send(count,id,destination,command,data);
    /* Define the expected response data for the START response and receive i
*/
    count = 6;
    id = 1;
    destination = SYSTEM;
    command = START;
    info_receive(id, destination, command, data);
} /**** end of info_start *****/

/*****
info_trans(color) char color; /*
*
* Send Set Transparent Color command and wait to receive
* the response.
*
{
    short int count;
    char id,destination,command,data[10];
    count = 7;
    id = 1;
    destination = VIDEO;
    command = TRANSPARENT;
    data [0] = color; /* color
    data [1] = color;
    info_send(count,id,destination,command,data);

    /* Define the expected response data for the TRANSPARENT response and re-
ceive*/
    count = 6;
    id = 1;
    destination = VIDEO;

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

    command = TRANSPARENT;
    info_receive(id, destination, command, data);
} /* end of info_trans */

/*****
info_wake_up(function_addr) int (*function_addr)(unsigned short, unsigned short);
unsigned short); */
*
* Send Wake-up command and wait to receive response.
*
{
    short int count;
    char id, destination, command, data[4];

    count = 9;
    id = 1;
    destination = SYSTEM;
    command = WAKEUP;

    wakeup_address.address = function_addr;
    /* assign address of the wakeup routine to the wakeup_address union.
*/
    data [0] = wakeup_address.addr_array [2]; /* low byte - segm */
    data [1] = wakeup_address.addr_array [3]; /* hi byte - segm */
    data [2] = wakeup_address.addr_array [0]; /* low byte - offs */
    data [3] = wakeup_address.addr_array [1]; /* hi byte - offs */
    info_send(count, id, destination, command, data);

    /* Define the expected response data for the WAKEUP response.
*/
    count = 6;
    id = 1;
    destination = SYSTEM;
    command = WAKEUP;
    info_receive(id, destination, command, data);
} /*end wake_up */

/*****
info_rom_speak(count, data) short int count; char data[];
/*
* Send ROM Speak command count is the number of bytes of data (only) */
{
    char id, destination, command;
    count = count + 5;
    id = 1;
    destination = ROMSPEECH;
    command = ROM_SPEAK;
    info_send(count, id, destination, command, data);

    /* Define the expected response data for the INITIALIZE response.
*/
    count = 6;
    id = 1;
    destination = ROMSPEECH;

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

    command = ROM_SPEAK;
    info_receive(id, destination, command, data);
} /*end info_rom_speak*/

/*****
info_init_video() /*
*
* Send VDP Initialize command and wait to receive response.
*
{
    short int count;
    char id,destination,command,data[1];
    count = 5;
    id = 1;
    destination = VDP1;
    command = VDP_INIT;
    info_send(count,id,destination,command,data);

    /* Define the expected response data for the INITIALIZE response.
*/
    count = 6;
    id = 1;
    destination = VDP1;
    command = VDP_INIT;
    info_receive(id, destination, command, data);
} /*end info_init_video*/

/*****
info_enable_audio() /*
*
* Send VDP Audio command to enable both VDP audio channels. Wait to
* receive response.
*
{
    short int count;
    char id,destination,command,data[1];

    count = 6;
    id = 1;
    destination = VDP1;
    command = VDP_AUDIO;
    data[0] = 0x03;
    info_send(count,id,destination,command,data);

    /* Define the expected response data for the VDP AUDIO response.
*/
    count = 6;
    id = 1;
    destination = VDP1;
    command = VDP_AUDIO;
    info_receive(id, destination, command, data);
} /* end info_enable audio */

```


COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/*****
info_match_frame(frame) short int frame; /*
*
* Send Match Frame command for frame
*
{
short int count;
char id,destination,command,data[10];
count = 7;
id = 1;
destination = VDP1;
command = MATCH_FRAME;
data[0] = frame >> 8; /* take high byte */
data[1] = frame & 0xFF; /* low byte */
info_send(count,id,destination,command,data);
} /*end match_frame */

/*****
info_play(frame,direct) short int frame; char direct; /* 0 forward, 1 backwar
*/
/*
* Send VDP Play command for frame and wait to receive Play response.
*
{
short int count;
char id,destination,command,data[3];
count = 8;
id = 1;
destination = VDP1;
command = PLAY;
data[0] = direct; /* play direction */

data[1] = frame >> 8; /*high byte */
data[2] = frame & 0xFF; /* low byte */

info_send(count,id,destination,command,data);

/* Define the expected response data for the VDP PLAY response.
*/
count = 6;
id = 1;
destination = VDP1;
command = PLAY;
info_receive(id, destination, command, data);
} /*end info_play */
/*****
info_from_to(frame,toframe) short int frame,toframe;

/*
* Send VDP command for from to play dont wait for response
*
{
short int count;

```

COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED

```

char id,destination,command,data[4];
count = 9;
id = 0;
destination = VDPl;
command = FROM_TO;
data[0] = frame >> 8; /*high byte */
data[1] = frame & 0xFF; /* low byte */
data[2] = toframe >> 8; /*high byte */
data[3] = toframe & 0xFF; /* low byte */
if (frame == 0) count = 5; /* no data, to cancel */
info_send(count,id,destination,command,data);
) /*end info_play */

/*****
info_load_key(count,data) short int count; char data[]; /*
*
* Send Load Key Table command, count is the length of data ,
* its format is key number,upper left row then column, lower right row
* then column, i.e. 5 bytes per key to set
*
{
char id,destination,command;
count = count + 5; /* for the command info*/
id = 1;
destination = TOUCH;
command = KEY_TABLE;
info_send(count,id,destination,command,data);

/* Define the expected response data for the LOAD KEYTABLE response.
*/
info_receive(id, destination, command, data);
} /*end load key*/
*****/
info_video_control() /*
*
* Send Video Control command to: enable video input 1, transparency,
* normal color palette and composite source 1; disable: interlace,
* auxiliary video output and composite source 2. Wait to receive response.
*
{
short int count;
char id,destination,command,data[10];
count = 7;
id = 1;
destination = VIDEO;
command = VIDEO_CNTRL;
data[0] = 0x10; /* Disable interlace, normal color */
/* palette, enable transparency, disable
/* auxiliary video output
data[1] = 0x80; /* enable composite source 1, disable */
/* source 2
info_send(count,id,destination,command,data);

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/* Define the expected response data for the VIDEO CONTROL response. */
count = 6;
id = 1;
destination = VIDEO;
command = VIDEO_CNTRL;
info_receive(id, destination, command, data);
} /*end video control */
/*****
info_touch(onoff) char onoff; /*
*
* Send Touch Control command to enable a touch in key table mode and to
* enable good beep. Wait to receive response.
* 1 for row/column, 2 for x/y, 3 for key table , 0 for off
{
short int count;
char id,destination,command,data[10];

count = 6;
id = 1;
destination = TOUCH;
command = TOUCH_CNTRL;
switch (onoff)
{
case 0 :
data[0] = 0x00;
break;
case 1 :
data[0] = 0xC9;
break;
case 2 :
data[0] = 0xC5;
break;
case 3 :
data[0] = 0xC3;
break;
} /* end switch */

info_send(count,id,destination,command,data); /* Define the expected resp
data for the TOUCH CONTROL response. */
id = 1;
destination = TOUCH;
command = TOUCH_CNTRL;
info_receive(id, destination, command, data);
} /*end info_touch */
*****/

info_still_frame(frame) /*displays a still frame */
short int frame;
{
short int count;
char id,destination,command,data[10];

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

count = 7;
id = 1;
destination = VDP1;
command = STILL_FRAME;
data[1] = frame >> 8; /*high byte */
data[2] = frame & 0xFF; /* low byte */
info_send(count,id,destination,command,data);
info_receive(id, destination, command, data);
) /*end still frame */

/*****/

int info_read_frame() /* returns the current frame number */
{
    short int count;
    char id,destination,command,data[10];
    int result;

    count = 5;
    id = 1;
    destination = VDP1;
    command = READ_FRAME;
    info_send(count,id,destination,command,data);
    info_receive(id, destination, command, data);
    result = (data[1]<<8) + data[2]; /*frame returned */
    return (result);
} /*end info_read_frame */

/*****/

info_display(onoff) /* show frame number on screen */
    char onoff; /* 1 is on, 0 is off */
{
    short int count;
    char id,destination,command,data[10];

    count = 6;
    id = 1;
    destination = VDP1;
    command = DISPLAY;
    data[0] = onoff;
    info_send(count,id,destination,command,data);
    info_receive(id, destination, command, data);
} /*end info_display */

/*****/
info_exit() /*spin down disc */
{
    short int count;
    char id,destination,command,data[10];

    count = 5;

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

id = 1;
destination = VDP1;
command = VDP_EXIT;
info_send(count,id,destination,command,data);
/* Define the expected response data for the VDP1 EXIT */
info_receive(id, destination, command, data);
} /* end info_exit */
/*****/

info_stop() /*
*
* Send Stop command and wait to receive response.
*
{
short int count;
char id,destination,command,data[1];
count = 5;
id = 1;
destination = SYSTEM;
command = STOP;
info_send(count,id,destination,command,data);

/* Define the expected response data for the STOP response.
*/
count = 6;
id = 1;
destination = SYSTEM;
command = STOP;
info_receive(id, destination, command, data);
} /*end info_stop */

/*****/
info_rec_match() /*receive matchframe function*/
{

short int count;
char id,destination,command;
char data[1];

/* Receive the Match Frame response. */
count = 6;
id = 1;
destination = VDP1;
command = MATCH_FRAME;
data [0] = 0x00;
info_receive(id, destination, command, data);
} /*end info_rec_match */

/*****/

int info_touch_res(data) /*receive touch response*/
char data[];
{

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

short int count;
char id,destination,command;
int key; /* valid in key mode*/

count = 5;
id = 00;
destination = TOUCH;
command = TOUCH_RESP;
info_receive(id, destination, command, data);
key = data[2];
if (data[1] != 2) key = 0; /* not in key mode */
return(key);
} /*end info_touch_res*/
/*****
info_sound_beep()
/*
* Send Touchscreen Sound Beep command to sound a good beep and wait to
* receive response.
*
{
short int count;
char id,destination,command;
char data[1];

count = 6;
id = 1;
destination = TOUCH;
command = SOUND_BEEP;
data [0] = 0x80; /* sound good beep */
info_send(count,id,destination,command,data);

/* Define the expected response data for the SOUND BEEP response.
count = 6;
id = 1;
destination = TOUCH;
command = SOUND_BEEP;
info_receive(id, destination, command, data);
} /*end sound_beep */

*****/
info_send (count,id,destination,command,indata) short int count; char
id,destination,command; unsigned char indata[];

/*****
* This routine sends a command buffer to the InfoWindow. After return from
* AVISND, avi.sndst is checked. If zero is found, return. If avi.sndst is
* a 1, call RECEIVE to receive any error responses, decipher the error
* response and abort. If avi.sndst is anything else, display a description
* of the error and abort.
*****/
{

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```
int cntr, datalen; union REGS inregs; union REGS outregs; struct SREGS segregs;
struct databuf sendbuf_ptr, recbuf_ptr; struct cntbuf troll_ptr; int intno; /*
interrupt number */ short int far *longptr;
```

```
/****** first place values in send buffer */ sendbuf_ptr.cc
= count; sendbuf_ptr.id = id; sendbuf_ptr.destination = destination;
sendbuf_ptr.command = command;
```

```
datalen = count - 5; /* amount of data */ if (datalen > 0) { /* place data
its section */
    for (cntr = 0; cntr < datalen; cntr++)
    {
        sendbuf_ptr.data[cntr] = indata[cntr];
    }
} /* end of datalen loop */
```

```
/****** now set up control buffer */
troll_ptr.kind = 0x0000; /*this is a send*/
longptr = &sendbuf_ptr.count;
troll_ptr.off_data = FP_OFF(longptr);
troll_ptr.seg_data = FP_SEG(longptr);
longptr = &avi.sndst;
troll_ptr.off_global = FP_OFF(longptr); /*first of global */
troll_ptr.seg_global = FP_SEG(longptr);
```

```
/****** now set up for interrupt *****/
```

```
/* place segment address of control block */
longptr = &troll_ptr.kind;
segregs.ds = FP_SEG(longptr); /* place offset address of control block */
inregs.x.dx = FP_OFF(longptr); /* place word in A register */
inregs.x.ax = 0x0107; /* issue the interrupt */
intno = 0x7F;
avi.sndst = 4; /*change it so it is a positive test */
int86x(intno, &inregs, &outregs, &segregs); /* interrupt */
```

```
/******check for error *****/
/* If avi.sndst is not zero, print out an error message and quit.
if (avi.sndst != 0x0000)
{
    printf ("\nAfter sending a command, variable avi.sndst indicates an er-
ror");
    switch (avi.sndst)
    {
        case 0x0001:
            printf("\n Unable to send code ");
            break;
        case 0x0002:
            printf ("\nAn unsupported Destination Code was specified.");
            break;
        case 0x0003:
            printf ("\nBuffer sent was of an invalid length
```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```
(%i)", sendbuf_ptr.count);
    break;
case 0x0004:
    printf ("\nAll transmit buffers are full");
    break;
case 0x0005:
    printf ("\nCommand rejected. Display is in an invalid state");
    break;
case 0x0006:
    printf ("\nResident PC code is not loaded");
    break;
} /* end switch */
exit(); /*stop the program */
) /* end if avi.sndst not zero */ ) /* end info_
```


COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

(C) SYNTHES 1987 ALL RIGHTS RESERVED          */ /*      COPYR
/*****
 * Library routines for using the cursor not! IBM InfoWindow.
 *
 *****/ #include <memory.h> #include
<dos.h>

/* global area to save keys loaded */ struct {
    int ulr; /* upper left row */
    int ulc; /* upper left col */
    int lrr; /* lower right row */
    int lrc; /* lower right col */ } keys[60]; char touch_type; /* 1 for
row/column, 2 for x/y, 3 for key table , 0 for off */

info_start() /* this routine need not do anything for the cursor*/
{
    return;
} /***** end of info_start *****/
/*****
info_trans(color) char color; /*
*
 * this routine need not do anything if there is no video system */
{
    return;
} /* end of info_trans */
/*****
info_wake_up(function_addr) int (*function_addr)(unsigned short, unsigned sho
unsigned short); /*
*
 * dont do anything - no video */
{
    return;
} /*end wake_up */
/*****
info_rom_speak(count,data) short int count; char data[];
/*
 * sorry, when no video - no speech*/
{
    return;
} /*end info_rom_speak*/
/*****
info_init_video() /*
*
 * dont do anything */
{
    return;
} /*end info_init_video*/
/*****
info_enable_audio() /* ;
*

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

* dont do anything */
{
    return;
} /* end info_enable audio */
/*****
info_match_frame(frame) short int frame; /*
*
* dont do anything */
{
    return;
} /*end match_frame */

/*****
info_play(frame,direct) short int frame; char direct; /* 0 forward, 1 backwar
*/ /*
* dont do anything */
{
    return;
} /*end info_play */
/*****
info_from_to(frame,toframe) short int frame,toframe; /*
*
* dont do anything */
{
    return;
} /*end info_play */
/*****
info_load_key(count,data) short int count; char data[]; /*
*
* Send Load Key Table command, count is the length of data ,
* its format is key number,upper left row then column, lower right row
* then column, i.e. 5 bytes per key to set
*
{
    int i;
    if (count > 300) return(1);
    for (i=0 ; i < count; i=i+5)
    {
        keys[data[i]].ulr = data[i+1];
        keys[data[i]].ulc = data[i+2];
        keys[data[i]].lrr = data[i+3];
        keys[data[i]].lrc = data[i+4];
    } /* end for */
    if (count == 0) { /* this is code to remove key table */
        for (i=0;i<60;i++)
        {
            keys[i].ulr = 0;
            keys[i].ulc = 0;
            keys[i].lrr = 0;
            keys[i].lrc = 0;
        }
    }
} /*end load key*/ ;
/*****

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

info_video_control() /*
*
do nothing */
{
return;
} /*end video control */
/*****
info_touch(onoff) char onoff;
/* 1 for row/column, 2 for x/y, 3 for key table , 0 for off
/* save this type */
{
touch_type = onoff;
} /*end info_touch */
/*****
info_still_frame(frame) /*displays a still frame */
short int frame;
/* dont do anything */
{
return;
} /*end still frame */
/***** i
info_read_frame() /* returns the current frame number */
{
/* dont do anything */
return(0);
} /*end info_read_frame */
/*****
info_display(onoff) /* show frame number on screen */
char onoff; /* 1 is on, 0 is off */
{
/* dont do anything */
return;
} /*end info_display */
/*****
info_exit() /*spin down disc */
/* dont do anything */
{
return;
} /* end info_exit */
/*****
info_stop() /*
*
* Send Stop command and wait to receive response.
*
{
/* dont do anything */
return;
} /*end info_stop */
/*****
info_rec_match() /*receive matchframe function*/
{
/* dont do anything */
} /*end info_rec_match */
/*****

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

int info_touch_res(data) /*receive touch response,uhm change to cursor */
/* assumes an egamode is in use */
char data[];
{
    int sw,i,x,y,area,x2,y2,key;
    unsigned int ix,iy;
    int maxx,maxy,row,col;
    int mode,type;
    float ratx,raty;
    /* get screen mode in current use */
    mode = getmode();
    switch (mode) {
        case 13:
            maxx = 320;
            maxy = 200;
            break;
        case 14:
            maxx = 640;
            maxy = 200;
            break;
        case 15:
            maxx = 640;
            maxy = 350;
            break;
        case 16:
            maxx = 640;
            maxy = 350;
            break;
        default:
            printf(" ega not in use ");
            return;
            break;
    } /* end mode switch */
    /* move cursor to middle if it is too far off */
    gcurget(&x,&y);
    if (x < 5 ) gcurloc(maxx/2,maxy/2);
    if (y < 5 ) gcurloc(maxx/2,maxy/2);
    if (x > maxx - 5 ) gcurloc(maxx/2,maxy/2);
    if (y > maxy - 5 ) gcurloc(maxx/2,maxy/2);
    gcurget(&x2,&y2);
    gcursize(11,11,14,10,0);
    /* get a cursor position */
    sw = 0; /* return on after keyhit */
    i=gcurmove(sw,&x,&y,&key);

    /* response based upon touch_type */
    data[0] = 0; /* no error */
    ratx = 255./maxx;
    raty = 255./maxy;
    ix = x*ratx; /* transfer from ega x to ibm x */
    iy = y*raty; /* transfer from ega y to ibm y */
    row=(iy-11)/9; /*transfer from ibm y to row */
    col=(ix-6)/3; /*transfer from ibm x to col */

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

switch (touch_type)
{
  case 1: /* row column */
    data[1]=8; /* mode returned */
    data[2]=row;
    data[3]=col;
    break;
  case 2: /*x-y return */
    data[1]=4; /* mode returned */
    data[2]=ix;
    data[3]=iy;
    break;
  case 3: /* key table return */
    data[1]=2; /* mode returned */
    /* find area */
    area = 0;
    i = 1;
    while (area == 0 && i < 60) {
      /* check if it is inside this one */
      if (row <= keys[i].lrr && row >= keys[i].ulr && col <=
keys[i].lrc && col >= keys[i].ulc)
      {
        area = i;
      } /* end if */
      i++;
    } /* end while */
    data[2]=area;
    data[3]=ix;
    data[4]=iy;
    break; /* last case of switch */
} /* end switch */
return (area);
} /*end info_touch_res*/
/*****
info_sound_beep()
/*
* Send Touchscreen Sound Beep command to sound a good beep and wait to
* receive response.
*
{
/* dont do anything */
return;
} /*end sound_beep */

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/*      COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED      */ #include
<stdio.h> #include <dos.h> #include <stdlib.h> #include <memory.h> #include
<conio.h> /***** */ int
part_halo_draw(name2,planum) /* clipping halo draw,direct to ega mem */
/* draw planum planes */ char name2[]; int planum;

{ unsigned add,mode ; /*ega ports*/ char color; int plane,s; FILE *stream; ex
int _clip,_cliplc,_cliplr,_clipuc,_clipur;

/* check picture file */ if ((stream = fopen(name2,"rb")) == NULL) return(1
fclose(stream); /* close it for now */

/* erase that box */ grrtulc(_clipuc,_clipur,_cliplc-_clipuc,_cliplr-_clipur,
/* first set write mode to 0*/ add = 0x3CE; mode = 0x3CF; outp(add,5);
outp(mode,0);

/* now set data mask to full*/ outp(add,0x8); outp(mode,0xFF);

/* now loop for each plane */ color = 1;

for ( plane=0; plane<planum; plane++) /* # planes needed*/
{
/* set the map mask for this plane */
add = 0x3C4; /* address for map mask */
mode = 0x3C5;
outp(add,2);
outp(mode,color);

if ((s=part_draw_plane(name2,plane,planum)) ==1) return(1); /* draw the
plane */
color = color << 1; /* next color */
}
return(0); } /* end of part_halo_draw */
/*****/

int part_draw_plane(name2,plane,planum) /* routine to draw a plane of data
char name2[]; int plane,planum; {

char far *plength,*pointer,*pmin,*pmax; static FILE *stream; unsigned char
line[512]; /* to hold line of data */ int datalen,numread,status; register
short int loop,count; /* grab variables from graphics package */ extern int
_clip,_cliplc,_cliplr,_clipuc,_clipur; int row,col;

count = 0; FP_SEG(plength) = 0xA000; FP_OFF(plength) = 80 * 350 ; /* number
bytes in plane */

if (plane == 0 ) {
/* open picture file */
if ((stream = fopen(name2,"rb")) == NULL) {
fprintf(stderr,"couldn't open file %s\n",name2);
return(1);
}
}

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

numread = fread(line,1,512,stream); /* read the first block */
if ((line[0] != 0x41) || (line[1] != 0x48) )/* sure dr halo */
{initgraf(3,0,0); /* back to text mode */
fprintf(stderr,"error - not a dr halo file ");
initgraf(3,0,0); /* set back to 80 column with color */
info_stop();/*stop it */
exit(1);}

/* mode of 640 X 350 16 colors */
count = 12; /* need to skip past header file */
} /* end if plane 0 */ if (plane != 0 ) numread = fread(line,1,512,stream);
/* just read in first line*/

/* set pointer to beginning of plane */

/* screen begins at a000 0000 */ FP_SEG(pointer) = 0xA000; FP_OFF(pointer)
0x0000;

FP_OFF(pmax) = _cliplr * 80 + _cliplc/8; /* lower right of clip boundary */
FP_SEG(pmax) = 0xA000;

if (plength < pmax) pmax = plength; /* use clipping boundary or end of plane

FP_SEG(pmin) = 0xA000; FP_OFF(pmin) = _clipur * 80 + _clipuc/8; /* upper left
clip boundary */

while(numread == 512 ) /* for each block */
{
while((count < 512) || (line[count] != 128)) /* for each byte */
{
if (line[count]==0) break; /*end of block */
if (line[count]==0x80) break; /*end of block */
datalen = line[count]&0x7F;
if(line[count]&0x80) /* duplication symbol */
{
count++;
for (loop=0;loop < datalen ; loop++)
{
if (pointer>pmin && pointer<pmax && line[count]!=0) *pointer
line[count];
pointer++;
} /* end looping */
count++; /* done with this data */
} /* end duplication loop */
else /* no duplication */
{
count++;
for (loop = 0 ; loop < datalen ;loop++)
{
if (pointer >pmin && pointer < pmax) *pointer = line[count];
pointer++;
count++; /*done with this data */
}
}
}
}

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

        } /* end looping */
    } /* end no duplications */
} /* end block */
numread = 0;
if (pointer < plength) { /* test for full screen */
    numread = fread(line,1,512,stream); /* read the next line */
    count = 0;
}
} /*end of plane */ if (plane == planum-1) {
status=fclose(stream);
if (status !=0) {
    fprintf(stderr,"error closing picture ");
    exit(1);
}
}
return; } /* end of part_draw_plane */
/*****
load_choice() /*this routine loads keys 40,41 with the return and next keys*/
{
    int ulc,ulr,lrc,lrr; /* clipping borders */
    short int zero;
    static char datal[15]={
        40,20,10,25,24,41,20,53,25,69,42,20,32,25,43};
    char data[40];
    ulc = 0;
    ulr = 0;
    lrr = 295;
    lrc = 640;
    zero = 0;
    info_load_key(zero,data); /* erase all keys first */
    strncpy(data,datal,15);
    info_load_key(15,data);
    clipsize(ulc,ulr,lrc,lrr); /* clip size,rest of screen */
    return; } /* end of load_choice */
/*****
set_choice
/*this routine sets up for the screen bottom*/
{
    int ulc,ulr,lrc,lrr; /* clipping borders */
    static char datal[15]={
        40,20,10,25,24,41,20,53,25,69,42,20,32,25,43};
    char data[40];
    ulc = 0;
    ulr = 252;
    lrr = 350;
    lrc = 640;
    info_load_key(0,data); /* erase all keys first */
    clipsize(ulc,ulr,lrc,lrr);
    part_halo_draw("c:\\clinical\\choice.pic",4);
    strncpy(data,datal,15);
    info_load_key(15,data);
    ulr = 0;
    lrr = 295;

```


COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

        clipsize(ulc,ulr,lrc,lrr); /* clip size,rest of screen */
        return; } /* end set_choice */
/***** mayexit(
/* might exit here, give user second chance */
{
    char data[10];
    int rs;
    info_touch(3); /*touch on */
    if ((rs=info_touch_res(data))==42)
    {
        initgraf(3,0,0); /* set back to 80 column with color */
        info_stop(); /*stop it */
        fcloseall(); /* close all files */
        exit(1);
    }
    info_touch(0); /* touch off */
    return;
} /*****/ int
cut_halo_draw(file,x,y) /*given x,y location draw a cut file there */ char
file[]; int x,y; {

FILE *stream; unsigned char line[512]; /* to hold line of data */ short int
length,width,height; int numread,status; int
dup,h,lg,count,loop,datalen,offy,offx; char color;

/* open file */ if ((stream = fopen(file,"rb")) == NULL) {
    fprintf(stderr,"couldn't open file %s\n",file);
    return(1);
} numread = fread(line,1,512,stream); /* read the first block */ if ((line
!= 0x00) || (line[5] != 0x00) ) /* sure dr halo */
{initgraf(3,0,0); /* back to text mode */
    fprintf(stderr,"error - not a dr halo file\n");
    return(1); }

/* the first 4 bytes relate to the width and height */ width = line[0]; heigh
line[2]; count = 6; /* towards start of data */ offx = 0; offy = 0; for
(h=height; h > 0 ;h--) /* draw each row as read */
{
    length = line[count];
    count_upd(&count,&numread,line,stream,&lg);
    count_upd(&count,&numread,line,stream,&lg);
    lg = count;
    while ( (count - lg) < length) { /* for these many bytes */

        datalen = line[count]&0x7F;
        if((dup=line[count])&0x80) /* duplication symbol */
        {
            count_upd(&count,&numread,line,stream,&lg);
            color = line[count]; /* use this data */
            count_upd(&count,&numread,line,stream,&lg);
            if (color ==0 ) { /* skip background color !! */
                for (loop=0;loop < datalen ; loop++)
                {
                    offx++;

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

        if ( offx >= width ) {
            offx = 0;
            offy++; }
    } /* end looping */
} /* end if no color */
if (color != 0) {
    for (loop=0; loop < datalen ; loop++)
    {
        dot(offx+x,offy+y,color);
        offx++;
        if ( offx >= width ) {
            offx = 0;
            offy++; }
    } /* end looping */
} /* end color */

} /* end duplication loop */
if(!(dup&0x80)) /* no duplication symbol */
{
    count_upd(&count,&numread,line,stream,&lg);
    for (loop = 0 ; loop < datalen ;loop++)
    {
        color = line[count]; /* use this data */
        count_upd(&count,&numread,line,stream,&lg);
        if (color!=0) dot(offx+x,offy+y,color);
        offx++;
        if ( offx >= width ) {
            offx = 0;
            offy++; }
    } /* end looping */
} /* end no duplications */
} /* end row while */
} /* end for height */
status = fclose(stream);
if (status != 0) printf("\n\n error closing stream\n");
return(0); } /* end of cut_halo_draw */

```

```

/*****/
count_upd(count,numread,line,stream,lg) /* used by routine cut_helo_draw to
update count and check if read necessary */

int *count,*numread; char line[]; FILE *stream; int *lg;
{
    (*count)++;
    if ((*count) >= (*numread)) { /*read in next line */
        *numread = fread(line,1,512,stream); /* read the first block */
        *lg = (*lg)-(*count); /* set back a few to take account of those read
        *count = 0; }
    return;
} /* end of count_upd */

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

/*      COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED      */ /* libra
of functions for manipulating windows for touchscreen */ /* assumptions : ega
initialized, touch screen set up, font 0 loaded */

#include <string.h> #include <malloc.h> #include <stdio.h>

/* global window parameters */ static struct {
    char *p;
    int ulc,ulr;
}window[10];

int wind_make(number,ulc,ulr,lrc,nlines) /*initialize a window */

int number; /* the window number being created ,from 0 to 9*/ int ulc,ulr,lrc
coordinates (high resolution apa) */ int nlines; /* number of lines (rows) fo
the window */

{
    int lrr,size;

    if (number > 9 || number < 0 ) return(1);
    lrr = ( (nlines) * 14) + ulr+10; /* 1 line per word */
    if (lrr > 350 || lrc > 640) {
        fprintf(stderr,"bad window coordinates");
        exit(1);
    }
    window[number].ulc=ulc;
    window[number].ulr=ulr;
    clipsize(ulc,ulr,lrc,lrr); /* set a clip boundary */
    /* allocate some memory*/
    size=(4+((( lrc-ulc)*4) +7) /8) * lrr-ulr);
    window[number].p = malloc(size);
    if (window[number].p == NULL) {
        fprintf(stderr,"error - window space not allocated ");
        return(1);
    }
    /* save that portion */
    get(ulc,ulr,lrc,lrr>window[number].p);
    /*erase that portion */
    grrtulc(ulc,ulr,lrc-ulc,lrr-ulr,0);
    /*make box */
    grrtul(ulc+1,ulr+1,lrc-ulc-1,lrr-ulr-1,15);
    return(0);
} /* end of wind_make */

/***** i
wind_line(number,string,line_no,touch_no,color) /* write a line to a window

int number; /*window number */ char string[]; /* string */ int line_no; /* l
number */ int touch_no; /*touch area number (or zero for none)*/ int color;

{
    char data[10];

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```

    if (number > 9 || number < 0 ) {
        fprintf(stderr,"window number not in range %i",number);
        return(1);
    }
    if (window[number].p==NULL) {
        fprintf(stderr,"window not initialized ");
        return(2);
    }
    /* write line to the window */
    fontsprt(0,string,window[number].ulc+5,window[number].ulr+14*(1+line_
no),0,color,0);
    /* set up the key-table for touch screen */
    if (touch_no > 0 && touch_no < 61) {
        data[0] = touch_no;
        data[1] = window[number].ulr/14+line_no;
        data[2] = window[number].ulc/8;
        data[3] = window[number].ulr/14+line_no+1;
        data[4] = strlen(string)+data[2];
        info_load_key(5,data);/*set up key table*/
    }
    return(0);
} /*end wind_line */

/******/ int
wind_close(number) /* close the window */

int number; /*window number */

{
    if (number > 9 || number < 0 ) return(1);
    if (window[number].p==NULL) return(1); /*not initialized */
    /* replace area */
    put(window[number].ulc,window[number].ulr,window[number].p,1);
    free(window[number].p);
    return(0);
} /*wind_close*/

/******/ int
wind_save(number,ulc,ulr,lrc,lrr) /*save a window - dont erase it */

int number; /* the window number being created ,from 0 to 9*/ int
ulc,ulr,lrc,lrr; /* coordinates (high resolution apa) */

{
    int size;

    if (number > 9 || number < 0 ) return(1);
    if (lrr > 350 || lrc > 640) {
        fprintf(stderr,"bad window coordinates");
        exit(1);
    }
    window[number].ulc=ulc;
    window[number].ulr=ulr;
    clipsize(ulc,ulr,lrc,lrr); /* set a clip boundary */
    /* allocate some memory*/

```

COPYRIGHT (C) SYNTHES 1987

ALL RIGHTS RESERVED

```
size=(4+((( lrc-ulc)*4) +7) /8) * lrr-ulr);
window[number].p = malloc(size);
    if (window[number].p == NULL) {
        fprintf(stderr,"error - window space not allocated ");
        return(1);
    }
/* save that portion */
get(ulc,ulr,lrc,lrr,window[number].p);
return(0);
} /* end of wind_save */
```

COPYRIGHT SYNTHES (U.S.A.) 1987
ALL RIGHTS RESERVED

Rule Group ADVISE-RULES

RULE002 [ADVISE-RULES]

If a relatively very strong system is important,
Then 1) there is weakly suggestive evidence (30%) that the suggested treatment for this orthopedic fracture is not CAST, and
2) there is weakly suggestive evidence (20%) that the suggested treatment for this orthopedic fracture is PLATE, and
3) there is weakly suggestive evidence (30%) that the suggested treatment for this orthopedic fracture is NAIL, and
4) there is weakly suggestive evidence (10%) that the suggested treatment for this orthopedic fracture is LAG SCREWS, and
5) there is weakly suggestive evidence (10%) that the suggested treatment for this orthopedic fracture is WIRE, and
6) there is weakly suggestive evidence (20%) that the suggested treatment for this orthopedic fracture is JOINT REPLACEMENT.

IF: STRONGER

THEN: TREATMENT != "CAST" CF 30 AND TREATMENT = "PLATE" CF 20 AND TREATMENT =
"NAIL"
CF 30 AND TREATMENT = "LAG SCREWS" CF 10 AND TREATMENT = "WIRE" CF 10
AND TREATMENT = "JOINT REPLACEMENT" CF 20

RULE003 [ADVISE-RULES]

If an immediate stability of the system is important,
Then 1) there is weakly suggestive evidence (16%) that the suggested treatment for this orthopedic fracture is I-FIX, and
2) there is weakly suggestive evidence (30%) that the suggested treatment for this orthopedic fracture is not CAST, and
3) it is definite (100%) that the suggested treatment for this orthopedic fracture is not TRACTION, and
4) there is weakly suggestive evidence (30%) that the suggested treatment for this orthopedic fracture is not PLATE, and
5) there is weakly suggestive evidence (17%) that the suggested treatment for this orthopedic fracture is NAIL, and
6) there is weakly suggestive evidence (10%) that the suggested treatment for this orthopedic fracture is LAG SCREWS, and
7) there is weakly suggestive evidence (10%) that the suggested treatment for this orthopedic fracture is WIRE, and
8) there is suggestive evidence (50%) that the suggested treatment for this orthopedic fracture is JOINT REPLACEMENT, and
9) there is weakly suggestive evidence (27%) that the suggested treatment for this orthopedic fracture is OPEN REDUCTION.

IF: IMMEDIATELY-STABLE

THEN: TREATMENT = "I-FIX" CF 16 AND TREATMENT != "CAST" CF 30 AND TREATMENT
!= "TRACTION" AND TREATMENT != "PLATE" CF 30 AND TREATMENT = "NAIL" CF
17 AND TREATMENT = "LAG SCREWS" CF 10 AND TREATMENT = "WIRE" CF 10 AND
TREATMENT = "JOINT REPLACEMENT" CF 50 AND TREATMENT = "OPEN REDUCTION"
CF 27

RULE004 [ADVISE-RULES]

 If a procedure that does not disrupt the soft tissue is important,
 Then 1) there is suggestive evidence (75%) that the suggested treatment for this orthopedic fracture is A-FRAME, and
 2) there is weakly suggestive evidence (30%) that the suggested treatment for this orthopedic fracture is I-FIX, and
 3) there is weakly suggestive evidence (30%) that the suggested treatment for this orthopedic fracture is CAST, and
 4) there is weakly suggestive evidence (30%) that the suggested treatment for this orthopedic fracture is TRACTION, and
 5) there is weakly suggestive evidence (30%) that the suggested treatment for this orthopedic fracture is not PLATE, and
 6) there is weakly suggestive evidence (17%) that the suggested treatment for this orthopedic fracture is not NAIL, and
 7) there is weakly suggestive evidence (20%) that the suggested treatment for this orthopedic fracture is not LAG SCREWS, and
 8) there is weakly suggestive evidence (17%) that the suggested treatment for this orthopedic fracture is not WIRE, and
 9) there is weakly suggestive evidence (30%) that the suggested treatment for this orthopedic fracture is not JOINT REPLACEMENT, and
 10) there is weakly suggestive evidence (27%) that the suggested treatment for this orthopedic fracture is not OPEN REDUCTION.

IF: NON-DISRUPTING

THEN: TREATMENT = "A-FRAME" CF 75 AND TREATMENT = "I-FIX" CF 30 AND TREATMENT = "CAST" CF 30 AND TREATMENT = "TRACTION" CF 30 AND TREATMENT != "PLATE" CF 30 AND TREATMENT != "NAIL" CF 17 AND TREATMENT != "LAG SCREWS" CF 20 AND TREATMENT != "WIRE" CF 17 AND TREATMENT != "JOINT REPLACEMENT" CF 30 AND TREATMENT != "OPEN REDUCTION" CF 27

RULE005 [ADVISE-RULES]

 If a relatively accurate result is important,
 Then 1) there is weakly suggestive evidence (13%) that the suggested treatment for this orthopedic fracture is I-FIX, and
 2) there is weakly suggestive evidence (0%) that the suggested treatment for this orthopedic fracture is CAST, and
 3) there is weakly suggestive evidence (17%) that the suggested treatment for this orthopedic fracture is not TRACTION, and
 4) there is weakly suggestive evidence (23%) that the suggested treatment for this orthopedic fracture is NAIL, and
 5) there is weakly suggestive evidence (30%) that the suggested treatment for this orthopedic fracture is LAG SCREWS, and
 6) there is weakly suggestive evidence (0%) that the suggested treatment for this orthopedic fracture is WIRE, and
 7) there is weakly suggestive evidence (17%) that the suggested treatment for this orthopedic fracture is not JOINT REPLACEMENT, and
 8) there is weakly suggestive evidence (20%) that the suggested treatment for this orthopedic fracture is not OPEN REDUCTION, and
 9) there is weakly suggestive evidence (30%) that the suggested treatment for this orthopedic fracture is PLATE.

IF: ACCURATE
 THEN: TREATMENT = "I-FIX" CF 13 AND TREATMENT = "CAST" CF 0 AND TREATMENT !=
 "TRACTION"
 " CF 17 AND TREATMENT = "NAIL" CF 23 AND TREATMENT = "LAG SCREWS" CF 30
 AND TREATMENT = "WIRE" CF 0 AND TREATMENT != "JOINT REPLACEMENT" CF 17
 AND TREATMENT != "OPEN REDUCTION" CF 20 AND TREATMENT = "PLATE" CF 30

RULE006 [ADVISE-RULES]

 If a number of procedures can be suggested is not known,
 Then it is definite (100%) that a number of procedures can be suggested.

IF: FINISHED IS NOTKNOWN
 THEN: FINISHED

RULE008 [ADVISE-RULES]

 If a relatively easy procedure is important,
 Then 1) there is weakly suggestive evidence (30%) that the suggested
 treatment for this orthopedic fracture is CAST, and
 2) there is weakly suggestive evidence (30%) that the suggested
 treatment for this orthopedic fracture is TRACTION, and
 3) there is weakly suggestive evidence (10%) that the suggested
 treatment for this orthopedic fracture is not ANGLE, and
 4) there is weakly suggestive evidence (10%) that the suggested
 treatment for this orthopedic fracture is not CONDYLAR, and
 5) there is weakly suggestive evidence (17%) that the suggested
 treatment for this orthopedic fracture is not NAIL.

IF: EASY
 THEN: TREATMENT = "CAST" CF 30 AND TREATMENT = "TRACTION" CF 30 AND TREATMENT
 != "ANGLE" CF 10 AND TREATMENT != "CONDYLAR" CF 10 AND TREATMENT != "
 NAIL" CF 17

RULE009 [ADVISE-RULES]

 If it is important to quickly determine the success of the treatment,
 Then 1) there is weakly suggestive evidence (13%) that the suggested
 treatment for this orthopedic fracture is I-FIX, and
 2) there is weakly suggestive evidence (10%) that the suggested
 treatment for this orthopedic fracture is not CAST, and
 3) there is weakly suggestive evidence (10%) that the suggested
 treatment for this orthopedic fracture is not TRACTION, and
 4) there is weakly suggestive evidence (10%) that the suggested
 treatment for this orthopedic fracture is PLATE, and
 5) there is weakly suggestive evidence (10%) that the suggested
 treatment for this orthopedic fracture is LAG SCREWS, and
 6) there is weakly suggestive evidence (7%) that the suggested treatment
 for this orthopedic fracture is WIRE, and
 7) there is weakly suggestive evidence (3%) that the suggested treatment
 for this orthopedic fracture is JOINT REPLACEMENT.

IF: FEEDBACKS

THEN: TREATMENT = "I-FIX" CF 13 AND TREATMENT != "CAST" CF 10 AND TREATMENT != "TRACTION" CF 10 AND TREATMENT = "PLATE" CF 10 AND TREATMENT = "LAG SCREWS" CF 10 AND TREATMENT = "WIRE" CF 7 AND TREATMENT = "JOINT REPLACEMENT" CF 3

RULE010 [ADVISE-RULES]

If a treatment that's versatile in its application is important,
Then 1) there is weakly suggestive evidence (3Z) that the suggested treatment for this orthopedic fracture is I-FIX, and
2) there is weakly suggestive evidence (7Z) that the suggested treatment for this orthopedic fracture is CAST, and
3) there is weakly suggestive evidence (7Z) that the suggested treatment for this orthopedic fracture is TRACTION, and
4) there is weakly suggestive evidence (1Z) that the suggested treatment for this orthopedic fracture is not PLATE, and
5) there is weakly suggestive evidence (3Z) that the suggested treatment for this orthopedic fracture is not NAIL, and
6) there is weakly suggestive evidence (3Z) that the suggested treatment for this orthopedic fracture is LAG SCREWS, and
7) there is weakly suggestive evidence (3Z) that the suggested treatment for this orthopedic fracture is WIRE, and
8) there is weakly suggestive evidence (3Z) that the suggested treatment for this orthopedic fracture is OPEN REDUCTION.

IF: VERSATILE

THEN: TREATMENT = "I-FIX" CF 3 AND TREATMENT = "CAST" CF 7 AND TREATMENT = "TRACTION" CF 7 AND TREATMENT != "PLATE" CF 1 AND TREATMENT != "NAIL" CF 3 AND TREATMENT = "LAG SCREWS" CF 3 AND TREATMENT = "WIRE" CF 3 AND TREATMENT = "OPEN REDUCTION" CF 3

RULE026 [ADVISE-RULES]

If information available from initial system queries is not known,
Then 1) call a DOS program named , and
2) it is definite (100Z) that information available from initial system queries.

IF: INPUT-READ IS NOT KNOWN

THEN: DOS-CALL "initit.bat" 18750 AND INPUT-READ

RULE032 [ADVISE-RULES]

If a procedure which allows for settling to occur is important,
Then there is weakly suggestive evidence (30Z) that the suggested treatment for this orthopedic fracture is DHS.

IF: ALLOWS-SETTLING

THEN: TREATMENT = "DHS" CF 30

Rule Group META-RULES

No rules defined

Rule Group PATIENT-RULES

RULE019 [PATIENT-RULES]

If 1) the age of the patient is greater than 70, or
 2) 1) the gender of the patient is FEMALE, and
 2) the age of the patient is greater than 50, or
 3) the number of years the patient is expected to live is < 10,
 Then it is definite (100%) that the medical age of the patient is OLD.

IF: VALUE AGE > 70 OR SEX = "FEMALE" AND (VALUE AGE) > 50 OR LIFE-EXPECT = "< 10"

THEN: MED-AGE = OLD

RULE020 [PATIENT-RULES]

If 1) it is risky to keep the patient convalescent, or
 2) the patient is unreliable and might not follow a post-op recovery plan,
 Then it is definite (100%) that recovery time is unavailable for this patient.

IF: CONVAL-RISK OR UNRELIABLE

THEN: RECOVERY-UNAVAILABLE

RULE021 [PATIENT-RULES]

If 1) the height of the patient in centimeters is greater than 198, or
 2) the weight of the patient in kilos is greater than 136,
 Then it is definite (100%) that the patient is very large.

IF: VALUE HEIGHT > 198 OR VALUE WEIGHT > 136

THEN: ODD-SIZE

RULE022 [PATIENT-RULES]

If 1) the height of the patient in centimeters is less than 198, and
 2) the weight of the patient in kilos is less than 136,
 Then it is definite (100%) that the patient is not very large.

IF: VALUE HEIGHT < 198 AND VALUE WEIGHT < 136

THEN: ' ODD-SIZE

RULE025 [PATIENT-RULES]

If information available from initial system queries,
 Then read data from the file patient.

IF: INPUT-READ

THEN: READ* FRAME "patient" QUOTE (AGE SEX WEIGHT HEIGHT LIFEDEC) TALLY

RULE027 [PATIENT-RULES]

 If the age of the patient is less than 18,
 Then it is definite (100%) that the medical age of the patient is YOUNG.

IF: VALUE AGE < 18
 THEN: MED-AGE = YOUNG

RULE031 [PATIENT-RULES]

 If the medical age of the patient is OLD,
 Then it is definite (100%) that it is risky to keep the patient convalescent.

IF: MED-AGE = OLD
 THEN: CONVAL-RISK

RULE034 [PATIENT-RULES]

 If 1) the medical age of the patient is OLD, or
 2) this patient has been confined to a bed for an extended period, or
 3) bone resorption has taken place due to disuse,
 Then there is suggestive evidence (75%) that this bone site is osteoporotic.

IF: MED-AGE = OLD OR BED-RIDDEN OR ATROPHIED
 THEN: OSTEOPOROTIC CF 75

RULE036 [PATIENT-RULES]

 If the mechanism of the injury is simple fall,
 Then it is definite (100%) that this bone site is osteoporotic.

IF: MECHANISM = "simple fall"
 THEN: OSTEOPOROTIC

RULE038 [PATIENT-RULES]

 If information available from initial system queries,
 Then read data from the file disease.

IF: INPUT-READ
 THEN: READ: FRAME "disease" QUOTE (HEALING OSTEOPOROTIC CONVAL-RISK
 SURGERY-INTOLERABLE UNRELIABLE) TALLY

RULE043 [PATIENT-RULES]

If this bone site is not osteoporotic,
Then it is definite (100%) that this patient has not been confined to a bed
for an extended period.

IF: ! OSTEOPOROTIC
THEN: ' BED-RIDDEN

RULE044 [PATIENT-RULES]

If 1) the age of the patient is less than 60, and
2) the injury severity score is less than 30,
Then there is suggestive evidence (70%) that the number of years the patient
is expected to live is > 10.

IF: VALUE AGE < 60 AND VALUE ISS < 30
THEN: LIFE-EXPECT = "> 10" OF 70

Rule Group PROSCONS-RULES

RULE011 [PROSCONS-RULES]

If the trauma involves bilateral fractures of weight bearing bones,
Then there is strongly suggestive evidence (90%) that a relatively very
strong system is important.

IF: BI-WEIGHT
THEN: STRONGER CF 90

RULE013 [PROSCONS-RULES]

If 1) the medical age of the patient is YOUNG, or
2) surgery is intolerable for this patient,
Then there is weakly suggestive evidence (25%) that a procedure that does not
disrupt the soft tissue is important.

IF: MED-AGE = YOUNG OR SURGERY-INTOLERABLE
THEN: NON-DISRUPTING CF 25

RULE014 [PROSCONS-RULES]

If 1) 1) there is evidence that this fracture is not inherently stable, and
2) this is a potentially stable fracture, or
2) the medical age of the patient is YOUNG,
Then there is suggestive evidence (50%) that a relatively accurate result is
important.

IF: INHERENTLY-STABLE IS THOUGHTNOT AND POTENTIALLY-STABLE OR MED-AGE = YOUNG
THEN: ACCURATE CF 50

RULE015 [PROSCONS-RULES]

If 1) the number of bone fragments is greater than 6, or
2) surgery is intolerable for this patient,
Then there is suggestive evidence (70%) that a relatively easy procedure is
important.

IF: VALUE NUMBER-PIECES > 6 OR SURGERY-INTOLERABLE
THEN: EASY CF 70

RULE017 [PROSCONS-RULES]

 If 1) the gender of the patient is FEMALE, and
 2) there is evidence that the medical age of the patient is not OLD,
 Then there is suggestive evidence (50%) that a relatively good cosmetic
 result is important.

IF: SEX = "FEMALE" AND MED-AGE IS THOUGHTNOT OLD
 THEN: COSMETIC CF 50

RULE024 [PROSCONS-RULES]

 If 1) the classification for the degree of openness of the fracture is
 greater than 1, or
 2) the damage to the soft tissue is greater than 1,
 Then there is strongly suggestive evidence (90%) that a procedure that does
 not disrupt the soft tissue is important.

IF: VALUE OPEN-CLASS > 1 OR VALUE SOFT-TISSUE-DAMAGE > 1
 THEN: NON-DISRUPTING CF 90

RULE030 [PROSCONS-RULES]

 If 1) recovery time is unavailable for this patient, or
 2) the trauma involves bilateral fractures of weight bearing bones,
 Then there is strongly suggestive evidence (90%) that an immediate stability
 of the system is important.

IF: RECOVERY-UNAVAILABLE OR BI-WEIGHT
 THEN: IMMEDIATELY-STABLE CF 90

RULE033 [PROSCONS-RULES]

 If this bone site is osteoporotic,
 Then there is strongly suggestive evidence (90%) that a procedure which
 allows for settling to occur is important.

IF: OSTEOPOROTIC
 THEN: ALLOWS-SETTLING CF 90

RULE035 [PROSCONS-RULES]

 If this fracture damaged an articular surface,
 Then there is strongly suggestive evidence (90%) that a relatively accurate
 result is important.

IF: ARTICULAR
 THEN: ACCURATE CF 90

Rule Group TRAUMA-RULES

RULE007 [TRAUMA-RULES]

If information available from initial system queries,
Then read data from the file trauma.

IF: INPUT-READ

THEN: READ+ FRAME "trauma" QUOTE (FRAC-CLASS ARTICULAR INHERENTLY-STABLE
NUMBER-PIECES POTENTIALLY-STABLE WEIGHT-BEARING TREATMENT TREATMENT
TREATMENT TREATMENT TREATMENT) TALLY

RULE039 [TRAUMA-RULES]

If information available from initial system queries,
Then read data from the file tissue.

IF: INPUT-READ

THEN: READ+ FRAME "tissue" QUOTE (EXTR NERV RESP ABDM CARD SKIN COMP ISS
SOFT-TISSUE-DAMAGE OPEN-CLASS) TALLY

RULE040 [TRAUMA-RULES]

If the extremity score from the health trauma index is less than 4,
Then it is definite (100%) that the trauma involves bilateral fractures of
weight bearing bones is not true.

IF: VALUE EXTR < 4

THEN: BI-WEIGHT

RULE041 [TRAUMA-RULES]

If 1) the patient is very large, or
2) the injury severity score is greater than 30,
Then there is suggestive evidence (60%) that a treatment that's versatile in
its application is important.

IF: ODD-SIZE OR VALUE ISS > 30

THEN: VERSATILE CF 60

RULE042 [TRAUMA-RULES]

 If 1) the injury severity score is greater than 30, or
 2) the classification for the degree of openness of the fracture is
 greater than 1,
 Then it is definite (100%) that the mechanism of the injury is high energy.

IF: VALUE ISS > 30 OR VALUE OPEN-CLASS > 1
 THEN: MECHANISM = "high energy"

User defined Functions

CLOSE-TEMP [FUNCTIONS]

 TRANSLATION: (this closes the temporary file)
 TYPE: ACTION
 TEMPLATE: (VALUE)
 SOURCE: (LAMBDA
 (FILE)
 (WRITE "last line in file " FILE)
 (CLOSE-OUTPUT-PORT FILE))

OPEN-TEMP [FUNCTIONS]

 TEMPLATE: (TEXT)
 TYPE: EXPRESSION
 TRANSLATION: (open a temporary file for output)
 SOURCE: (LAMBDA
 (TEXT)
 (OPEN-OUTPUT-FILE TEXT))

WRITE-TEMP [FUNCTIONS]

 TEMPLATE: (VALUE VALUE)
 TYPE: ACTION
 TRANSLATION: (write out a test line to a file)
 SOURCE: (LAMBDA
 (FILE OUTP)
 (BEGIN
 (WRITE OUTP FILE)
 (NEWLINE FILE)))

Parameter Group ADVISE-PARMS

ACCURATE [ADVISE-PARMS]

 TRANSLATION: (a relatively accurate result is important)
 DEFAULT: (NO)
 TYPE: YES/NO
 USED-BY: RULE005
 UPDATED-BY: RULE014 RULE035

ALLOWS-SETTLING [ADVISE-PARMS]

 TRANSLATION: (a procedure which allows for settling to occur is important)
 TYPE: YES/NO
 USED-BY: RULE032
 UPDATED-BY: RULE033

COSMETIC [ADVISE-PARMS]

 TRANSLATION: (a relatively good cosmetic result is important)
 DEFAULT: (NO)
 TYPE: YES/NO
 UPDATED-BY: RULE017

EASY [ADVISE-PARMS]

 TRANSLATION: (a relatively easy procedure is important)
 TYPE: YES/NO
 DEFAULT: (NO)
 USED-BY: RULE008
 UPDATED-BY: RULE015

FEEDBACKS [ADVISE-PARMS]

 TRANSLATION: (it is important to quickly determine the success of the
 treatment)
 DEFAULT: (0*TRUE)
 TYPE: YES/NO
 USED-BY: RULE009

FINISHED [ADVISE-PARMS]

 TRANSLATION: (a number of procedures can be suggested)
 ACTIVE-VALUE: (DO-ALL
 (SET-VALUE OPENED
 (E
 (OPEN-TEMP "temper.txt")))
 (WRITE-TEMP
 (VALUE-OF OPENED)
 (VAL FRAME TREATMENT NOTRACEFLG))
 (CLOSE-TEMP
 (VALUE-OF OPENED))
 (DOS-CALL "" "debrief.bat" 18740))
 TYPE: YES/NO
 USED-BY: RULE006
 UPDATED-BY: SREFMARK RULE006

IMMEDIATELY-STABLE [ADVISE-PARMS]

 TRANSLATION: (an immediate stability of the system is important)
 TYPE: YES/NO
 DEFAULT: (NO)
 USED-BY: RULE003
 UPDATED-BY: RULE030 SREFMARK

INPUT-READ [ADVISE-PARMS]

 TRANSLATION: (information available from initial system queries)
 TYPE: YES/NO
 USED-BY: RULE026 RULE007 RULE025 RULE038 RULE039
 UPDATED-BY: SREFMARK RULE026

NON-DISRUPTING [ADVISE-PARMS]

 TRANSLATION: (a procedure that does not disrupt the soft tissue is important
)
 DEFAULT: (NO)
 TYPE: YES/NO
 USED-BY: RULE004
 UPDATED-BY: RULE013 RULE024

STRONGER [ADVISE-PARMS]

 TRANSLATION: (a relatively very strong system is important)
 DEFAULT: (NO)
 TYPE: YES/NO
 USED-BY: RULE002
 UPDATED-BY: RULE011

TREATMENT [ADVISE-PARMS]

 TRANSLATION: (the suggested treatment for this orthopedic fracture)
 LEGALVALUES: TEXT
 TYPE: MULTIVALUED
 UPDATED-BY: RULE010 RULE005 RULE009 RULE004 RULE002 RULE003 RULE032 RULE008
 RULE007

VERSATILE [ADVISE-PARMS]

 TRANSLATION: (a treatment that's versatile in its application is important)
 DEFAULT: (NO)
 TYPE: YES/NO
 USED-BY: RULE010
 UPDATED-BY: RULE041

Parameter Group FRAMETYPES

ADVISE [FRAMETYPES]

 TRANSLATION: (The treatment procedure suggestion)
 IDENTIFIER: "ADVISE-"
 RULEGROUPS: (ADVISE-RULES)
 PARMGROUP: ADVISE-PARMS
 GOALS: (INPUT-READ TREATMENT FINISHED)
 OFFSPRING: (PROSCONS)

PATIENT [FRAMETYPES]

 TRANSLATION: (the personality of the patient)
 PROMPTER: (patient factors being considered)
 PARENTS: (PROSCONS ADVISE)
 PARMGROUP: PATIENT-PARMS
 RULEGROUPS: (PATIENT-RULES)
 IDENTIFIER: "PATIENT-"

PROSCONS [FRAMETYPES]

 TRANSLATION: (the pros and cons which will determine the treatment of the fracture)
 PROMPTER: (treatment factors being established)
 GOALS: (STRONGER IMMEDIATELY-STABLE NON-DISRUPTING ACCURATE EASY VERSATILE FEEDBACKS COSMETIC)
 PARENTS: (ADVISE)
 PARMGROUP: PROSCONS-PARMS
 RULEGROUPS: (PROSCONS-RULES)
 IDENTIFIER: "PROSCONS-"
 OFFSPRING: (TRAUMA PATIENT)

TRAUMA [FRAMETYPES]

 TRANSLATION: (the personality of the trauma)
 PROMPTER: (trauma factors being considered)
 PARENTS: (PROSCONS ADVISE)
 PARMGROUP: TRAUMA-PARMS
 RULEGROUPS: (TRAUMA-RULES)
 IDENTIFIER: "TRAUMA-"

Parameter Group PATIENT-PARMS

AGE (PATIENT-PARMS)

TRANSLATION: (the age of the patient)
 PROMPT: YES
 EFFECT: POSITIVE-NUMBER
 RANGE: (1 110)
 TYPE: SINGLEVALUED
 USED-BY: RULE027 RULE019 RULE044
 UPDATED-BY: RULE025

ATROPHIED (PATIENT-PARMS)

TRANSLATION: (bone resorption has taken place due to disuse)
 TYPE: YES/NO
 USED-BY: RULE034

BED-RIDDEN (PATIENT-PARMS)

TRANSLATION: (this patient has been confined to a bed for an extended period
)
 PROMPT: YES
 TYPE: YES/NO
 USED-BY: RULE034
 UPDATED-BY: RULE043

CONVAL-RISK (PATIENT-PARMS)

TRANSLATION: (it is risky to keep the patient convalescent)
 PROMPT: YES
 TYPE: YES/NO
 USED-BY: RULE020
 UPDATED-BY: RULE031 RULE038

HEALING (PATIENT-PARMS)

TRANSLATION: (soft-tissue healing is an area of concern for this patient)
 PROMPT: (no)
 TYPE: YES/NO
 UPDATED-BY: RULE038

HEIGHT (PATIENT-PARMS)

TRANSLATION: (the height of the patient in centimeters)
 PROMPT: YES
 EFFECT: POSITIVE-NUMBER
 RANGE: (30 230)
 TYPE: SINGLEVALUED
 USED-BY: RULE021 RULE022
 UPDATED-BY: RULE025

LIFE-EXPECT [PATIENT-PARMS]

 TRANSLATION: (the number of years the patient is expected to live)
 PROMPT: YES
 EXPECT: ("< 10" "> 10")
 TYPE: SINGLEVALUED
 USED-BY: RULE019
 UPDATED-BY: RULE044

LIFEDEC [PATIENT-PARMS]

 TRANSLATION: (the general lifestyle or occupation of the patient reflecting
 use of the fracture site)
 PROMPT: YES
 TYPE: SINGLEVALUED
 EXPECT: ("ACTIVE" "SEDENTARY" "VIGOROUS")
 UPDATED-BY: RULE025

NAME [PATIENT-PARMS]

 TRANSLATION: (the name of the patient)
 PROMPT: YES
 TYPE: SINGLEVALUED

UNRELIABLE [PATIENT-PARMS]

 TRANSLATION: (the patient is unreliable and might not follow a post-op
 recovery plan)
 PROMPT: YES
 CERTAINTY-FACTOR-RANGE: UNKNOWN
 TYPE: YES/NO
 USED-BY: RULE020
 UPDATED-BY: RULE038

WEIGHT [PATIENT-PARMS]

 TRANSLATION: (the weight of the patient in kilos)
 PROMPT: YES
 EXPECT: POSITIVE-NUMBER
 RANGE: (5 180)
 TYPE: SINGLEVALUED
 USED-BY: RULE021 RULE022
 UPDATED-BY: RULE025

Parameter Group PROSCONS-PARMS

ARTICULAR [PROSCONS-PARMS]

 TRANSLATION: (this fracture damaged an articular surface)
 TYPE: YES/NO
 USED-BY: RULE035
 UPDATED-BY: RULE007 SREFMARK

BI-WEIGHT [PROSCONS-PARMS]

 TRANSLATION: (the trauma involves bilateral fractures of weight bearing bones)
 PROMPT: YES
 TYPE: YES/NO
 USED-BY: RULE011 RULE030
 UPDATED-BY: RULE040

INHERENTLY-STABLE [PROSCONS-PARMS]

 TRANSLATION: (this fracture is inherently stable)
 TYPE: YES/NO
 USED-BY: RULE014
 UPDATED-BY: RULE007 SREFMARK

ISS [PROSCONS-PARMS]

 TRANSLATION: (the injury severity score)
 TYPE: SINGLEVALUED
 USED-BY: RULE041 RULE042 RULE044
 UPDATED-BY: RULE039

MECHANISM [PROSCONS-PARMS]

 TRANSLATION: (the mechanism of the injury)
 PROMPT: YES
 EXPECT: ("simple fall" "low energy" "high energy")
 TYPE: SINGLEVALUED
 USED-BY: RULE036
 UPDATED-BY: RULE042

MED-AGE [PROSCONS-PARMS]

 TRANSLATION: (the medical age of the patient)
 TYPE: SINGLEVALUED
 EXPECT: (YOUNG OLD)
 USED-BY: RULE013 RULE031 RULE014 RULE017 RULE034
 UPDATED-BY: RULE027 RULE019

NUMBER-PIECES [PROSCONS-PARMS]

 TRANSLATION: (the number of bone fragments)
 TYPE: SINGLEVALUED
 USED-BY: RULE015
 UPDATED-BY: RULE007 SREFMARK

ODD-SIZE [PROSCONS-PARMS]

 TRANSLATION: (the patient is very large)
 PROMPT: YES
 TYPE: YES/NO
 USED-BY: RULE041
 UPDATED-BY: RULE021 RULE022

OPEN-CLASS [PROSCONS-PARMS]

 TRANSLATION: (the classification for the degree of openness of the fracture)
 PROMPT: YES
 EXPECT: POSITIVE-NUMBER
 TYPE: SINGLEVALUED
 RANGE: (0 3)
 USED-BY: RULE024 RULE042
 UPDATED-BY: RULE039

OSTEOPOROTIC [PROSCONS-PARMS]

 TRANSLATION: (this bone site is osteoporotic)
 PROMPT: YES
 TYPE: YES/NO
 DEFAULT: (YES -30)
 USED-BY: RULE033 RULE043
 UPDATED-BY: RULE036 RULE038 RULE034

POTENTIALLY-STABLE [PROSCONS-PARMS]

 TRANSLATION: (this is a potentially stable fracture)
 TYPE: YES/NO
 USED-BY: RULE014
 UPDATED-BY: RULE007 SREFMARK

RECOVERY-UNAVAILABLE [PROSCONS-PARMS]

 TRANSLATION: (recovery time is unavailable for this patient)
 DEFAULT: (NO)
 TYPE: YES/NO
 USED-BY: RULE030
 UPDATED-BY: RULE020

SEX [PROSCONS-PARMS]

TRANSLATION: (the gender of the patient)

PROMPT: YES

EXPECT: (MALE FEMALE)

TYPE: SINGLEVALUED

USED-BY: RULE017 RULE019

UPDATED-BY: RULE025

SOFT-TISSUE-DAMAGE [PROSCONS-PARMS]

TRANSLATION: (the damage to the soft tissue)

PROMPT: YES

RANGE: (0 3)

EXPECT: POSITIVE-NUMBER

TYPE: SINGLEVALUED

USED-BY: RULE024

UPDATED-BY: RULE039

SURGERY-INTOLERABLE [PROSCONS-PARMS]

TRANSLATION: (surgery is intolerable for this patient)

PROMPT: YES

CERTAINTY-FACTOR-RANGE: POSITIVE

TYPE: YES/NO

USED-BY: RULE013 RULE015

UPDATED-BY: RULE038

WEIGHT-BEARING [PROSCONS-PARMS]

TRANSLATION: (this is a weight bearing bone)

TYPE: YES/NO

UPDATED-BY: RULE007 SREFMARK

Parameter Group TRAUMA-PARMS

ABDM [TRAUMA-PARMS]

TRANSLATION: (the abdominal cavity score from the health trauma index)
 TYPE: SINGLEVALUED
 UPDATED-BY: RULE039

CARD [TRAUMA-PARMS]

TRANSLATION: (the cardiovascular system score from the health trauma index)
 TYPE: SINGLEVALUED
 UPDATED-BY: RULE039

COMP [TRAUMA-PARMS]

TRANSLATION: (the complications score from the health trauma index)
 TYPE: SINGLEVALUED
 UPDATED-BY: RULE039

EXTR [TRAUMA-PARMS]

TRANSLATION: (the extremity score from the health trauma index)
 TYPE: SINGLEVALUED
 USED-BY: RULE040
 UPDATED-BY: RULE039

FRAC-CLASS [TRAUMA-PARMS]

TRANSLATION: (the classification code of the fracture)
 TYPE: SINGLEVALUED
 UPDATED-BY: RULE007 SREFMARK

NERV [TRAUMA-PARMS]

TRANSLATION: (the nervous system score from the health trauma index)
 TYPE: YES/NO
 UPDATED-BY: RULE039

RESP [TRAUMA-PARMS]

TRANSLATION: (the respiratory system score from the health trauma index)
 TYPE: SINGLEVALUED
 UPDATED-BY: RULE039

SKIN [TRAUMA-PARMS]

TRANSLATION: (the skin and subcutaneous tissue score from the health trauma index)
 TYPE: SINGLEVALUED
 UPDATED-BY: RULE039

System parameters

ADVISE-PARMS [PARMGROUPS]

 VALUE: ACCURATE ALLOWS-SETTLING COSMETIC EASY FEEDBACKS FINISHED
 IMMEDIATELY-STABLE INPUT-READ NON-DISRUPTING STRONGER TREATMENT
 VERSATILE

FRAMETYPES [PARMGROUPS]

 VALUE: PROSCONS TRAUMA PATIENT ADVISE

PATIENT-PARMS [PARMGROUPS]

 VALUE: AGE ATROPHIED BED-RIDDEN CONVAL-RISK HEALING HEIGHT LIFE-EXPECT
 LIFECC NAME UNRELIABLE WEIGHT

PROSCONS-PARMS [PARMGROUPS]

 VALUE: ARTICULAR BI-WEIGHT INHERENTLY-STABLE ISS MECHANISM MED-AGE
 NUMBER-PIECES ODD-SIZE OPEN-CLASS OSTEOPOROTIC POTENTIALLY-STABLE
 RECOVERY-UNAVAILABLE SEX SOFT-TISSUE-DAMAGE SURGERY-INTOLERABLE
 WEIGHT-BEARING

TRAUMA-PARMS [PARMGROUPS]

 VALUE: ABDOM CARD COMP EXTR FRAC-CLASS NERV RESP SKIN

System parameters

ADVISE-RULES [RULEGROUPS]

SVAL: (THE ADVISE)

FRAME: (ADVISE)

VALUE: RULE002 RULE003 RULE004 RULE005 RULE006 RULE008 RULE009 RULE010
RULE026 RULE032

META-RULES [RULEGROUPS]

VALUE:

PATIENT-RULES [RULEGROUPS]

FRAME: (PATIENT)

SVAL: (THE PATIENT)

VALUE: RULE019 RULE020 RULE021 RULE022 RULE025 RULE027 RULE031 RULE034
RULE036 RULE038 RULE043 RULE044

PROSCONS-RULES [RULEGROUPS]

FRAME: (PROSCONS)

SVAL: (THE PROSCONS)

VALUE: RULE011 RULE013 RULE014 RULE015 RULE017 RULE024 RULE030 RULE033
RULE035

TRAUMA-RULES [RULEGROUPS]

FRAME: (TRAUMA)

SVAL: (THE TRAUMA)

VALUE: RULE007 RULE039 RULE040 RULE041 RULE042

Variables

DOMAIN [VARIABLES]

VALUE: "Orthopedic Advisor"

OPENED [VARIABLES]

VALUE: 0

COPYRIGHT (C) SYNTHES 1987. ALL RIGHTS RESERVED.

	PROCEDUR.INS						
	138,EXTERNAL		X-FIX				A
-FRAME	138,EXTERNAL		X-FIX	EXT	0	0.	I
-FRAME	138,EXTERNAL		X-FIX		0	0.	C
-FRAME	138,EXTERNAL		X-FIX		0	0.	P
LASTIC	138,EXTERNAL		CAST		0	0.	P
LASTER	138,EXTERNAL		CAST		0	0.	P
	138,EXTERNAL		TRACTION		0	0.	
	138,INTERNAL		IMPLANT		0	0.	P
LATE	138,INTERNAL	STRAIGHT	IMPLANT	STRAT	0	0.	P
LATE	138,INTERNAL	2 STRAIGHT	IMPLANT		0	0.	P
LATE	138,INTERNAL	DHS	IMPLANT	DHS	2390043200.		P
LATE	138,INTERNAL	DCS	IMPLANT	DCS	1296028800.		P
LATE	138,INTERNAL	ANGLE	IMPLANT	ANGLE	0	0.	P
LATE	138,INTERNAL	CONDYLAR	IMPLANT	CONDY	0	0.	P
LATE	138,INTERNAL	BUTRESS	IMPLANT	BUTT1	0	0.	P
LATE	138,INTERNAL	2 BUTRESS	IMPLANT	BUTT2	0	0.	N
AIL	138,INTERNAL	UNIVERSAL	IMPLANT	UNAIL	012960.		N
AIL	138,INTERNAL	ENDER'S	IMPLANT		0	0.	N
AIL	138,INTERNAL	LOCKING	IMPLANT	UNAIL	012960.		L
AG SCREWS	138,INTERNAL		IMPLANT		0	0.	L
AG SCREWS	138,INTERNAL	+ ANGLE PLATE	IMPLANT	ANGLE	0	0.	L
AG SCREWS	138,INTERNAL	+ STRAIGHT PLATE	IMPLANT	STRAT	0	0.	L
AG SCREWS	138,INTERNAL	+ CONDYLAR PLATE	IMPLANT	CONDY	0	0.	L
AG SCREWS	138,INTERNAL	+ BUTRESS PLATE	IMPLANT	BUTT1	0	0.	W
IRE	138,INTERNAL		IMPLANT		0	0.	H
IP PROSTHESIS	138,INTERNAL		JOINT REPLACEMENT	HIP	0	0.	K
NEE REPLACEMENT	138,INTERNAL		JOINT REPLACEMENT		0	0.	
	138,INTERNAL		OPEN REDUCTION		0	0.	

COPYRIGHT (C) SYNTHES 1987. ALL RIGHTS RESERVED.

PROCSUGG.INS

```

79,31A1.1    ENDER'S
. 79,31A1.3    ANGLE
. 79,31A1.XXXXXDHS
. 79,31A2.1    ANGLE
. 79,31A2.2    DHS
. 79,31A2.3    DHS
. 79,31A2.3    HIP PROSTHESIS
. 79,31A3.3    DHS
. 79,31A3.XXXXXANGLE
. 79,31AXXXXXXXHIP PROSTHESIS
. 79,31B2.1    DHS
. 79,31B2.XXXXXLAG SCREWS
. 79,31B3.1    LAG SCREWS
. 79,31BXXXXXXANGLE
. 79,31BXXXXXXDHS
. 79,31BXXXXXXHIP PROSTHESIS
. 79,31CXXXXXXHIP PROSTHESIS
. 79,31CXXXXXXLAG SCREWS
. 79,32A1.1    LAG SCREWS
. 79,32A1.1    + ANGLE PLATE
. 79,32A2.1    LAG SCREWS
. 79,32A3.1    + ANGLE PLATE
. 79,32A3.1    LAG SCREWS
. 79,32A3.1    + ANGLE PLATE
. 79,32B1.1    + ANGLE PLATE
. 79,32B2.1    + ANGLE PLATE
. 79,32B3.1    + ANGLE PLATE
. 79,32C1.1    + ANGLE PLATE
. 79,32C2.1    + ANGLE PLATE
. 79,32C3.1    + ANGLE PLATE
. 79,32C1.1    TRACTION
. 79,32C2.1    TRACTION
. 79,32C3.1    TRACTION
. 79,32A1.2    UNIVERSAL
. 79,32A1.2    LAG SCREWS
. 79,32A2.2    UNIVERSAL
. 79,32A2.2    LAG SCREWS
. 79,32A2.2    + STRAIGHT PLATE
. 79,32A3.2    UNIVERSAL
. 79,32B1.2    UNIVERSAL
. 79,32B1.2    STRAIGHT
. 79,32B2.2    UNIVERSAL
. 79,32B2.2    STRAIGHT
. 79,32B3.2    UNIVERSAL
. 79,32C1.2    2 STRAIGHT
. 79,32C1.2    LOCKING
. 79,32C2.2    2 STRAIGHT
. 79,32C2.2    UNIVERSAL
. 79,32C3.2    2 STRAIGHT
. 79,32C3.2    LOCKING
. 79,32A1.3    UNIVERSAL

```

COPYRIGHT (C) SYNTHES 1987. ALL RIGHTS RESERVED.

- . 79,32A1.3 LAG SCREWS
- . 79,32A2.3 UNIVERSAL
- . 79,32A2.3 LAG SCREWS
- . 79,32A3.3 UNIVERSAL
- . 79,32B1.3 UNIVERSAL
- . 79,32B1.3 + STRAIGHT PLATE
- . 79,32B2.3 UNIVERSAL
- . 79,32B2.3 + STRAIGHT PLATE
- . 79,32B3.3 UNIVERSAL
- . 79,32C1.3 LOCKING
- . 79,32C2.3 UNIVERSAL
- . 79,32C3.3 LOCKING
- . 79,32XX.XXXXXX-FIX
- . 79,33A1.1 LAG SCREWS
- . 79,33A1.2 + CONDYLAR PLATE
- . 79,33A1.3 + CONDYLAR PLATE
- . 79,33A1.3 LOCKING
- . 79,33A2.XXXXXX+ CONDYLAR PLATE
- . 79,33A3.XXXXXXCONDYLAR
- . 79,33B1.XXXXXXLAG SCREWS
- . 79,33B1.XXXXXX+ CONDYLAR PLATE
- . 79,33B1.XXXXXXDCS
- . 79,33B2.XXXXXX+ BUTRESS PLATE
- . 79,33B3.XXXXXXLAG SCREWS
- . 79,33C1.XXXXXX+ CONDYLAR PLATE
- . 79,33C1.XXXXXXDCS
- . 79,33C2.XXXXXX+ BUTRESS PLATE
- . 79,33C2.XXXXXXDCS
- . 79,33C3.XXXXXX+ BUTRESS PLATE
- . 79,33C3.XXXXXX2 BUTRESS
- . 79,33C3.XXXXXXKNEE REPLACEMENT
- . 79,33C3.XXXXXX-FIX

Page No. 97 not being used.

COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED			
139,3	Femur	.	
139,31	Proximal Femur	.	
139,31A	Trochanteric Region	.	
139,31A1	Pertrochanteric w/ single extension into medial	.	cortex, lateral
intact		.	
139,31A1.1	above or through lesser trochanter	.	
139,31A1.2	impacted	.	
139,31A1.3	w/ 3rd posterior fragment	.	
139,31A2	Pertrochanteric with 2 or more fracture lines	.	extending medi
ally		.	
139,31A2.1	w/o posterior fragment	.	
139,31A2.2	w/ 4th posterosuperior fragment	.	
139,31A2.3	comminution	.	
139,31A3	intertrochanteric, lateral and medial cortices	.	fractured
139,31A3.1	oblique fracture line (reverse fracture)	.	
139,31A3.2	transverse fracture line	.	
139,31A3.3	comminuted w/ 3rd third medial fragment	.	
139,31B	Femoral Neck	.	
139,31B1	femoral neck in abduction	.	
139,31B1.1	posterior greater than 20 degrees	.	
139,31B1.2	posterior displacement 10 to 20 degrees	.	
139,31B1.3	posterior displacement under 10 degrees	.	
139,31B2	femoral neck w/ vertical fracture line	.	
139,31B2.1	basilar neck	.	
139,31B2.2	medial neck	.	
139,31B2.3	subcapital	.	
139,31B3	femoral neck in adduction	.	

139,31B3.1	moderate varus	.
139,31B3.2	moderate translation	.
139,31B3.3	great displacement	.
139,31C	Femoral Head	.
139,31C1	separation only	.
139,31C1.1	torn ligament	.
139,31C1.2	small fragment and torn ligament	.
139,31C1.3	large fragment and torn ligament	.
139,31C2	rotation only	.
139,31C2.1	posterior-superior	.
139,31C2.2	anterior-superior	.
139,31C2.3	superio-lateral	.
139,31C3	combination	.
139,31C3.1	rotation and separation	.
139,31C3.2	separation and neck fracture	.
139,31C3.3	rotation and neck fracture	.
139,32	Femoral Shaft	.
139,32A	Simple Fracture	.
139,32A1	Simple spiral	.
139,32A1.1	subtrochanteric	.
139,32A1.2	midshaft	.
139,32A1.3	distal shaft	.
139,32A2	Simple oblique	.
139,32A2.1	subtrochanteric	.
139,32A2.2	midshaft	.
139,32A2.3	distal shaft	.
139,32A3	Simple transverse	.

139,32A3.1	subtrochanteric	.
139,32A3.2	midshaft	.
139,32A3.3	distal shaft	.
139,32B	Chip	.
139,32B1	torsional intact	.
139,32B1.1	subtrochanteric	.
139,32B1.2	midshaft	.
139,32B1.3	distal shaft	.
139,32B2	flexion intact	.
139,32B2.1	subtrochanteric	.
139,32B2.2	midshaft	.
139,32B2.3	distal shaft	.
139,32B3	fragmented	.
139,32B3.1	subtrochanteric	.
139,32B3.2	midshaft	.
139,32B3.3	distal shaft	.
139,32C	Complex	.
139,32C1	spiral comminuted	.
139,32C1.1	2 intermediary fragments	.
139,32C1.2	3 intermediary fragments	.
139,32C1.3	more than 3 intermediary fragments	.
139,32C2	complex bifocal	.
139,32C2.1	one intermediary fragment	.
139,32C2.2	one intermediary fragment and a chip	.
139,32C2.3	two intermediary fragments	.
139,32C3	non spiral comminuted	.
139,32C3.1	2-3 intermediary fragments	.

139,32C3.2	local comminution	.
139,32C3.3	extended comminution	.
139,33	Distal Femur	.
139,33A	Extrarticular	.
139,33A1	Simple	.
139,33A1.1	epicondylar chip	.
139,33A1.2	oblique	.
139,33A1.3	transverse	.
139,33A2	with a chip	.
139,33A2.1	1 large fragment	.
139,33A2.2	laterally fragmented	.
139,33A2.3	medially fragmented	.
139,33A3	comminuted	.
139,33-A3.1	2 large fragments	.
139,33A3.2	metaphysis comminuted	.
139,33A3.3	diaphysis and metaphysis comminuted	.
139,33B	Partial articular	.
139,33B1	unicondylar lateral	.
139,33B1.1	simple intercondylar	.
139,33B1.2	simple condylar	.
139,33B1.3	multiple fragments	.
139,33B2	unicondylar medial	.
139,33B2.1	simple intercondylar	.
139,33B2.2	simple condylar	.
139,33B2.3	multiple fragments	.

139,33B3	frontal	.	
139,33B3.1	anterio-lateral	osteochondral	.
139,33B3.2	unicondylar posterior	.	
139,33B3.3	bicondylar posterior	.	
139,33C	Total articular	.	
139,33C1	intercondylar simple with supracondylar simple	.	
139,33C1.1	T or Y slightly displaced	.	
139,33C1.2	T or Y very displaced	.	
139,33C1.3	T epiphysis	.	
139,33C2	intercondylar simple with supracondylar multiple	fragments	.
139,33C2.1	1 supracondylar fragment	.	
139,33C2.2	fragmented on one side	.	
139,33C2.3	comminuted	.	
139,33C3	complex multiple	fragments	.
139,33C3.1	sole intermediate	articular fragment	.
139,33C3.2	comminuted epiphysis -	metaphysis	.
139,33C3.3	comminuted epiphysis -	metaphysis - diaphysis	.

COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED

```

                                AOPC.BAT
echo off REM this routine starts PCPLUS ,queries user for options:
REM      developement environment?
REM      data to ram disk?
REM      touchscreen or cursor ?
REM      data and consulting on ram disk?
REM
REM *****
REM no switches or just an s means a simple run of the program
REM remove old error file erase error.tmp

if %1A==A goto simple if %1==s goto simple if %1==S goto simple
if %5A==A goto err if %1==C goto choicel

    echo DEVELOPE
    set pcpath=c:\pcplus
    set pcdrive=c:
    set datapath=c:\btrieve\orthobas
    set datadrive=c:
    set dafpath=c:\btrieve\orthobas
    set dafdrive=c:
    set btpath=c:\btrieve
    %pcdrive%
    cd %pcpath%
    copy c:\btrieve\orthobas\initit.bat initit.bat >> error.tmp
    copy c:\btrieve\orthobas\debrief.bat debrief.bat >> error.tmp
    goto choicell :choicel
    echo Consult
    set pcpath=c:\oarun
    set pcdrive=c:
    set dafpath=c:\oarun
    set dafdrive=c:
    set btpath=c:\oarun
    set datapath=c:\oarun
    set datadrive=c: :choicell if NOT %2==C goto choice2
    echo converge
    command /c c:\oarun\converge.bat :choice2 if NOT %3==F goto
choice3
    echo full use of ram disk
    set pcpath=e:\pcs
    set pcdrive=e:
    md %pcpath% >> error.tmp
    copy c:\oarun\*. * %pcpath% >> error.tmp :choice3 REM
***** if %4==C set interac=%4 if
%4==C echo cursor if %4==T echo touchscreen if %4==T set interac=%4

REM start mouse driver %pcpath%\mousesys >> error.tmp
%pcpath%\m_scheme >> error.tmp REM
***** if NOT %5==R goto choice4
    echo ram disk used for procedure data
    set datapath=e:\pcs
    set datadrive=e:
    %datadrive%

```

COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED

```

md %datapath% >> error.tmp :choice4 REM
***** :runit %pcdrive% cd
%pcpath% pause pc.bat ext goto end

:simple REM ***** REM this is
for no extra memory, ram drives or touchscreen, cursor is used echo
SIMPLE USAGE cd c:\oarun c: set interac=S REM start mouse driver
mousesys >> error.tmp m_scheme >> error.tmp pc.bat goto end

:err echo the correct format is 'aopc' (simple usage - with
cursor ) echo or 'aopc v w x y z' echo where v is a
C(consult) or D(develope) or S(simple) or a ? for this listing echo
w is a C(component upgrade) or N(no) echo x is an F(full
consulting on ram disk ) or N(no) echo y is a C(cursor ) or
T(touchscreen) echo z is a R(ram disk for data) or N(no) :end

INITIT.BAT

echo off
REM batch file for initial infromation gathering
REM this program creates files in the pcpath directory,
REM to be used by PCPLUS there
REM the system parameter interac determines touchscreen or REM not (C)
REM instead of touch-screen if %interac%==S goto simple if
%interac%==s goto simple REM complex directory switching
%btpath%\btrieve /P:1024 >>error.tmp

%dafdrive% cd %dafpath%

if %interac%==C goto cursor if %interac%==c goto cursor

initial.exe goto after :cursor initcur.exe goto after :after copy
trauma.rd %pcpath%\trauma.rd >> error.tmp copy patient.rd
%pcpath%\patient.rd >> error.tmp copy tissue.rd %pcpath%\tissue.rd >>
error.tmp copy disease.rd %pcpath%\disease.rd >> error.tmp
%btpath%\butil -stop %pcdrive% cd %pcpath%

goto end REM here everything is on one directory, cursor usage :simple
btrieve /P:1024 >>error.tmp initcur.exe butil -stop goto end

:end Rem finished with batch file

FINAL.BAT
echo off REM bath file for final infromation gathering
if %interac%==S goto simple if %interac%==s goto simple

%btpath%\btrieve /M:64/P:512/B:64 >> error.tmp %datadrive% cd
%datapath% copy %pcpath%\temper.txt temper.txt >> error.tmp copy
%pcpath%\trauma.rd trauma.rd >> error.tmp copy %pcpath%\patient.rd
patient.rd >> error.tmp copy %pcpath%\tissue.rd tissue.rd >>
error.tmp copy %pcpath%\disease.rd disease.rd >> error.tmp

copy %dafpath%\procedur.dat procedur.tmp >> error.tmp

```

COPYRIGHT (C) SYNTHES 1987 ALL RIGHTS RESERVED

```
if %interac%==C goto cursor if %interac%==c goto cursor

%dafpath%\final.exe goto after :cursor %dafpath%\fincur.exe goto after
:after %btpath%\butil -stop %pcdrive% cd %pcpath% goto end

:simple REM simple usage, cursor, no extra memory or directories
btrieve /M:64/P:512/B:64 >> error.tmp copy procedur.dat procedur.tmp >>
error.tmp fincur.exe butil -stop goto end

: end REM finished with final batch file
```


COPYRIGHT (C) SYNTHES 1987. ALL RIGHTS RESERVED

database: PREEEXIST							
61,CARDIOVASCULAR	VASC INSUFFICIENCY	0	0	0	50	0.	
61,CARDIOVASCULAR	ANEMIA	0	0	0	50	0.	
61,CARDIOVASCULAR	ARTERIALSCLEROSIS	0	0	0	50	0.	
61,CARDIOVASCULAR	ORGAN TRANSPLANT	30	50	0	0	0.	
61,CARDIOVASCULAR	OTHER	0	0	0	0	0.	
61,DIGESTIVE	OTHER	0	0	0	0	0.	
61,ENDOCRINE	DIABETES	0	0	0	50	0.	
61,ENDOCRINE	OTHER	0	0	0	0	0.	
61,INTEGUMENTARY	MELENOMA	30	0	0	0	0.	
61,INTEGUMENTARY	OTHER	0	0	0	0	0.	
61,LYMPHATIC	LYMPHOMA	0	0	0	50	0.	
61,LYMPHATIC	OTHER	0	0	0	0	0.	
61,MUSCULAR	POLIO MYELITIS	0	30	0	0	0.	
61,MUSCULAR	OTHER	0	0	0	0	0.	
61,NERVOUS	PARALYSIS	0	40	0	0	30.	
61,NERVOUS	STROKE	0	30	0	0	80.	
61,NERVOUS	POST POLIO	0	30	0	0	20.	
61,NERVOUS	MENINGITIS	0	0	0	80	0.	
61,NERVOUS	ALCOHOLIC	0	0	0	0	50.	
61,NERVOUS	ALZHEIMERS	0	0	0	0	80.	
61,NERVOUS	BLINDNESS	0	0	0	0	20.	
61,NERVOUS	OTHER	0	0	0	0	0.	
61,REPRODUCTIVE	OTHER	0	0	0	0	0.	
61,RESPIRATORY	HEAVY SMOKER	20	0	0	0	0.	
61,RESPIRATORY	TUBERCULOSIS	0	20	0	0	0.	
61,RESPIRATORY	EMPHYSEMA	0	0	30	0	0.	
61,RESPIRATORY	OTHER	0	0	0	0	0.	
61,SKELETAL	PRIMARY TUMOR	0	70	0	0	0.	
61,SKELETAL	METASTASIS	0	70	0	0	0.	
61,SKELETAL	OTHER	0	0	0	0	0.	
61,URINARY	OTHER	0	0	0	0	0.	

COPYRIGHT (C) SYNTHES 1987. ALL RIGHTS RESERVED

database: HTRAUMA

61, ABDOMINAL	No injury	.
61, ABDOMINAL	mild abdominal wall, flank or back pain	.
61, ABDOMINAL	acute flank, back or abdominal pain	.
61, ABDOMINAL	one of: minor liver, sm bowel, spleen,
61, ABDOMINAL	2 major: rupture liver, bladder, colon,
61, ABDOMINAL	2 severe: crush liver, major vascular	.
61, CARDIOVASCULAR	No injury	.
61, CARDIOVASCULAR	< 10% blood volume (bv) loss	.
61, CARDIOVASCULAR	10-20 % bv loss, dec. skin perfusion	.
61, CARDIOVASCULAR	20-30 % bv loss, urine (>30cc) tamponade	.
61, CARDIOVASCULAR	30-40 % bv loss, urine (<10cc) tamponade	.
61, CARDIOVASCULAR	40-50 % bv loss, restless, agitated, coma	.
61, CARDIOVASCULAR	50 % bv loss, coma, cardiac arrest	.
61, COMPLICATIONS	No complications	.
61, COMPLICATIONS	subq. wound infect, atelectasis tmp <38.5	.
61, COMPLICATIONS	major wound infect, atelectasis tmp >38.5	.
61, COMPLICATIONS	i.p. abscess, pneumonia, anuria, oliguria	.
61, COMPLICATIONS	septicemia, empyema, renal failure	.
61, COMPLICATIONS	above w/ low bp, gi bleed, resp arrest	.
61, COMPLICATIONS	above > 6 wks, coma > 6wks	.
61, EXTREMITIES	No injury	.
61, EXTREMITIES	Minor sprns and fxs no long bones	.
61, EXTREMITIES	simple fxs: not femur or pelvis	.
61, EXTREMITIES	simple femur or pelvis, multiple moderate	.
61, EXTREMITIES	fxs two major, unstable pelvis, limb loss	.
61, EXTREMITIES	fxs two severe, multiple major	.
61, NERVOUS	No injury	.
61, NERVOUS	head trauma w/ or w/o scalp lacer	.
61, NERVOUS	head trauma w/ brief coma (<15')	.
61, NERVOUS	cerebral inj. w/ coma (+15'), skull fx	.
61, NERVOUS	cerebral inj. w/ coma (+60'), neuro fdgs	.
61, NERVOUS	as above w/ dilated fixed pupils	.
61, NERVOUS	as above w/ no response up to 24 hrs	.
61, RESPIRATORY	No Injury	.
61, RESPIRATORY	Chest discomfort	.
61, RESPIRATORY	Simple rib or sternal fracture	.
61, RESPIRATORY	1st or multi-rub fx, hemothorax	.
61, RESPIRATORY	open chest wounds, flail chest	.
61, RESPIRATORY	acute resp. failure (cyanosis)	.
61, SKIN	No injury	.
61, SKIN	< 5 % burn, abrasions, contusions, lacer.	.
61, SKIN	5-15% burn, extensive contusions, avulsns	.
61, SKIN	15-30% burn, avulsions	.
61, SKIN	30-45% burn, avulsions entire leg or arm	.
61, SKIN	45-60% burn (3rd degree)	.
61, SKIN	60% + burn (3rd degree)	.

COPYRIGHT (C) SYNTHES 1987. ALL RIGHTS RESERVED.

```

database: CLASSEXP
25,31A1.1    noyes01yesyes.
25,31A1.2    noyes01yesyes.
25,31A1.3    noyes02yesyes.
25,31A2.1    no no02yesyes.
25,31A2.2    no no03yesyes.
25,31A2.3    no no10 noyes.
25,31A3.1    no no01yesyes.
25,31A3.2    no no01yesyes.
25,31A3.3    no no03yesyes.
25,31B1.1    no no01yesyes.
25,31B1.2    no no01yesyes.
25,31B1.3    no no01yesyes.
25,31B2.1    no no01yesyes.
25,31B2.2    no no01yesyes.
25,31B2.3    no no01 noyes.
25,31B3.1    no no01 noyes.
25,31B3.2    no no01yesyes.
25,31B3.3    no no02yesyes.
25,31C1.1    yesyes01yesyes.
25,31C1.2    yesyes01yesyes.
25,31C1.3    yesyes01yesyes.
25,31C2.1    yesyes01yesyes.
25,31C2.2    yesyes01yesyes.
25,31C2.3    yesyes01yesyes.
25,31C3.1    yes no01yesyes.
25,31C3.2    yes no01yesyes.
25,31C3.3    yes no01yesyes.
25,32A1.1    noyes01yesyes.
25,32A1.2    noyes01yesyes.
25,32A1.3    noyes01yesyes.
25,32A2.1    noyes01yesyes.
25,32A2.2    noyes01yesyes.
25,32A2.3    noyes01yesyes.
25,32A3.1    noyes01yesyes.
25,32A3.2    noyes01yesyes.
25,32A3.3    noyes01yesyes.
25,32B1.1    noyes02yesyes.
25,32B1.2    noyes02yesyes.
25,32B1.3    noyes02yesyes.
25,32B2.1    noyes03yesyes.
25,32B2.2    noyes03yesyes.
25,32B2.3    noyes03yesyes.
25,32B3.1    noyes04yesyes.
25,32B3.2    noyes04yesyes.
25,32B3.3    noyes04yesyes.
25,32C1.1    no no02yesyes.
25,32C1.2    no no02yesyes.
25,32C1.3    no no02yesyes.
25,32C2.1    noyes10 noyes.
25,32C2.2    noyes10 noyes.
25,32C2.3    noyes10 noyes.

```

COPYRIGHT (C) 1987 SYNTHES. ALL RIGHTS RESERVED.

25,32C3.1	no no10 noyes.
25,32C3.2	no no10 noyes.
25,32C3.3	no no10 noyes.
25,33A1.1	no no01yesyes.
25,33A1.2	no no01yesyes.
25,33A1.3	no no02yesyes.
25,33A2.1	no no01yesyes.
25,33A2.2	no no01yesyes.
25,33A2.3	no no02yesyes.
25,33A3.1	no no10yesyes.
25,33A3.2	no no10yesyes.
25,33A3.3	no no10 noyes.
25,33B1.1	no no01yesyes.
25,33B1.2	no no01yesyes.
25,33B1.3	no no02yesyes.
25,33B2.1	no no01yesyes.
25,33B2.2	no no01yesyes.
25,33B2.3	no no02yesyes.
25,33B3.1	yesyes01yesyes.
25,33B3.2	no no01yesyes.
25,33B3.3	no no01yesyes.
25,33C1.1	no no02yesyes.
25,33C1.2	no no02yesyes.
25,33C1.3	no no03yesyes.
25,33C2.1	no no10yesyes.
25,33C2.2	no no10yesyes.
25,33C2.3	no no10 noyes.
25,33C3.1	no no10yesyes.
25,33C3.2	no no10 noyes.
25,33C3.3	no no10 noyes.

12 **EUROPEAN PATENT APPLICATION**

21 Application number: 88112881.3

51 Int. Cl.⁵: G06F 15/42

22 Date of filing: 08.08.88

30 Priority: 13.08.87 US 85511

43 Date of publication of application:
22.02.89 Bulletin 89/08

84 Designated Contracting States:
CH DE FR GB LI

85 Date of deferred publication of the search report:
21.03.90 Bulletin 90/12

71 Applicant: **SYNTHE AG**
Grabenstrasse 15
CH-7002 Chur(CH)

72 Inventor: **Dormond, Kenneth**
30 Deer Run Lane
Malvern, Pennsylvania 19355(US)
Inventor: **Friedman, Robert J.**
1100 West Chester Pike, Apt. D 8
Paoli, Pennsylvania 19382(US)

74 Representative: **Lusuardi, Werther Giovanni**
Dr. Lusuardi AG Kreuzbühlstrasse 8
CH-8008 Zürich(CH)

54 **Expert system.**

57 An expert system which provides one or more suggested treatments for a patient with physical trauma is disclosed. The system includes a computing device having a memory, a plurality of data bases in the memory, an application program and an inference engine program. The data bases include graphical illustrations of different types of physical trauma, and a knowledge base which contains treatment information. The application program is executed in the computing device and interactively displays a series of screens including at least some of the graphical illustrations, to elicit responses from the user concerning the specific type of physical trauma and specific characteristics of the patient. The inference engine program, which is also executed in the computing device, uses the knowledge base and information related to the responses elicited from the user, for selecting one or more suggested treatments. The application program presents the suggested treatments to the user after execution of the inference engine program.

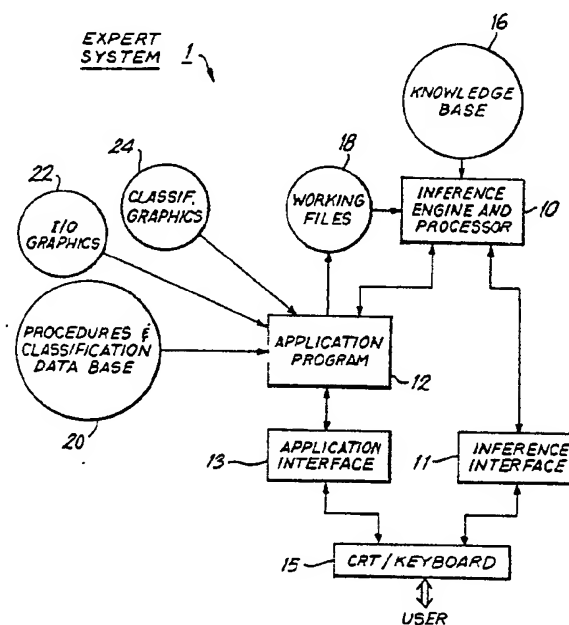


FIG. 1



EP 88 11 2881

DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.4)
A	EXPERT SUPPORT SYSTEM IN GOVERNMENT SYMPOSIUM 22-24 October 1986, IEEE Comput. Soc. Press, XIII + 466pp, pages 183-188, Melean, VA, USA; H. C. GABLER et al.: "An expert Support System for the Classification of Crash Victim Trauma" * whole document * ---	1,2	G 06 F 15/42
A	PROCEEDINGS OF THE 1986 IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS 14-17 October 1986, pages 552-556, Atlanta, Georgia, USA; L.-M. FU: "Integration of phenomenological and Fundamental Knowledge in Diagnostic Expert Systems" * whole document * ---	1,2	
A	PROCEEDINGS OF THE 1986 IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS 14-17 October 1986, pages 205-210, Atlanta, Georgia, USA; N. F. EZQUERRA et al.: "Development of an expert system for interpreting medical images" * whole document * -----	1,2	
			TECHNICAL FIELDS SEARCHED (Int. Cl.4)
			G 06 F 15/42 G 06 F 15/40 G 06 F 9/44 A 61 B 10/00
The present search report has been drawn up for all claims			
Place of search BERLIN		Date of completion of the search 23-11-1989	Examiner DURAND J.
CATEGORY OF CITED DOCUMENTS			
X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure P : intermediate document		T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons ----- & : member of the same patent family, corresponding document	